The sip:provider CE Handbook mr3.5.1

Sipwise GmbH

<support@sipwise.com>

Contents

1 Introduction

1.1 About this Document

This document describes the architecture and the operational steps to install, operate and modify the Sipwise sip:provider CE.

In the various chapters, it describes the system architecture, the installation and upgrade procedures and the initial configuration steps to get your first users online. It then dives into advanced preference configurations like rewrite rules, call blockings, call forwards etc.

There is a description of the customer self-care interface, how to configure the billing system and how to provision the system via the provided APIs.

Finally it describes the internal configuration framework, the network configuration and gives hints about tweaking the system for security and performance.

1.2 Getting Help

1.2.1 Community Support

We have set up the *spce-user* mailing list, where questions are answered on a best-effort basis and discussions can be started with other community users.

1.2.2 Commercial Support

If you need professional help setting up and maintaining the sip:provider CE, send an email to support@sipwise.com.

Sipwise also provides training and commercial support for the platform. Additionally, we offer a migration path to the sip:provider PRO appliance, which is the commercial, carrier-grade version of the sip:provider CE. If the user base grows on the CE, this will allow operators to migrate seamlessly to a highly available and scalable platform with defined service level agreements, phone support and on-call duty. Please visit www.sipwise.com for more information on commercial offerings.

1.3 What is the sip:provider CE?

The sip:provider CE is a SIP based Open Source Class5 VoIP soft-switch platform providing rich telephony services. It offers a wide range of features to end users (call forwards, voicemail, conferencing, call blocking, click-to-dial, call-lists showing near-realtime accounting information etc.), which can be configured by them using the customer-self-care web interface. For operators, it offers a fully web-based administrative panel, allowing them to configure users, peerings, billing profiles etc., as well as viewing real-time statistics of the system. For tight integration into existing infrastructures, it provides a powerful REST API.

The sip:provider CE can be installed in a few steps within a couple of minutes and requires no knowledge about configuration files of specific software components.

1.4   What is inside the sip:provider CE?

Opposed to other free VoIP software, the sip:provider CE is not a single application, but a whole software platform, the Sipwise NGCP (Sipwise Next Generation Communication Platform), which is based on Debian GNU/Linux.

Using a highly modular design approach, the NGCP leverages popular open-source software like MySQL, Apache, Catalyst, Kamailio, SEMS, Asterisk etc. as its core building blocks. These blocks are glued together using optimized and proven configurations and work-flows and are complemented by building blocks developed by Sipwise to provide fully-featured and easy to operate VoIP services.

After downloading and starting the installer, it will fetch and install all the required Debian packages from the relevant Debian repositories. The installed applications are managed by the NGCP Configuration Framework, which allows to change system parameters in a single place, so administrators don't need to have any knowledge of the dozens of different configuration files of the different packages. This provides a very easy and bullet-proof way of operating, changing and tweaking the otherwise quite complex system.

Once configured, integrated web interfaces are provided for both end users and administrators to use the sip:provider CE. By using the provided provisioning and billing APIs, it can be integrated tightly into existing OSS/BSS infrastructures to optimize work-flows.

1.5   Who should use the sip:provider CE?

The sip:provider CE is specifically tailored to companies and engineers trying to start or experiment with a fully-featured SIP based VoIP service without having to go through the steep learning curve of SIP signalling, integrating the different building blocks to make them work together in a reasonable way and implementing the missing components to build a business on top of that.

In the past, creating a business-ready VoIP service included installation and configuration of SIP software like Asterisk, OpenSER, Kamailio etc., which can get quite difficult when it comes to implementing advanced features. It required to implement different web interfaces, billing engines and connectors to existing OSS/BSS infrastructure. These things are now obsolete due to the CE, which covers all these requirements.

2 Platform Architecture

The sip:provider CE platform is one single node running all necessary components of the system. The components are outlined in the following figure:
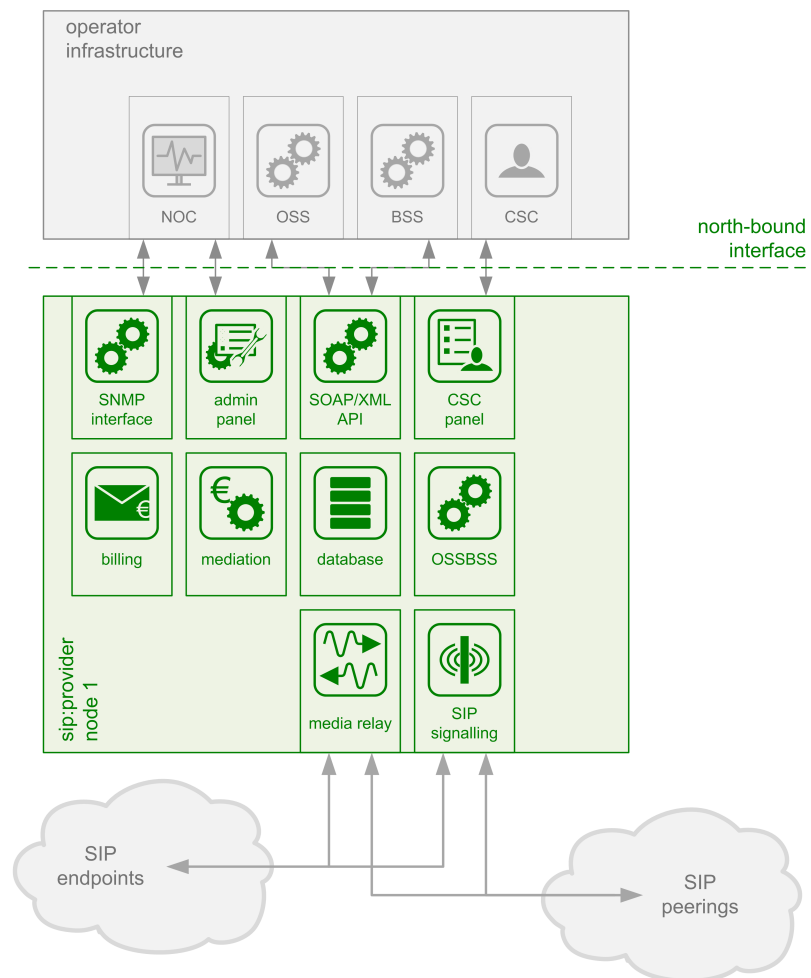
Figure 1: Architecture Overview

The main building blocks of the sip:provider CE are:

- SIP Signaling and Media Relay

- Provisioning

- Mediation and Billing

2.1 SIP Signaling and Media Relay

In SIP-based communication networks, it is important to understand that the signaling path (e.g. for call setup and tear-down) is completely independent of the media path. On the signaling path, the involved endpoints negotiate the call routing (which user

calls which endpoint, and via which path - e.g. using SIP peerings or going through the PSTN - the call is established) as well as the media attributes (via which IPs/ports are media streams sent and which capabilities do these streams have - e.g. video using H.261 or Fax using T.38 or plain voice using G.711). Once the negotiation on signaling level is done, the endpoints start to send their media streams via the negotiated paths.

2.1.1 SIP and Media Elements

The components involved in SIP and Media on the sip:provider CE are shown in the following figure:
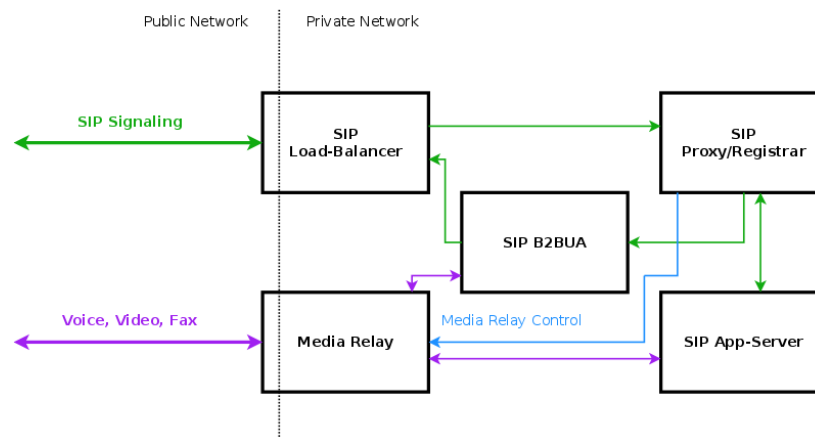
Figure 2: SIP and Media Relay Components

SIP Load-Balancer

The SIP load-balancer is a Kamailio instance acting as ingress and egress point for all SIP traffic to and from the system. It's a high-performance SIP proxy instance based on Kamailio and is responsible for sanity checks of inbound SIP traffic. It filters broken SIP messages, rejects loops and relay attempts and detects denial-of-service and brute-force attacks and gracefully handles them to protect the underlying SIP elements. It also performs the conversion of TLS to internal UDP and vice versa for secure signaling between endpoints and the sip:provider CE, and does far-end NAT traversal in order to enable signaling through NAT devices.

The load-balancer is the only SIP element in the system which exposes a SIP interface to the public network. Its second leg binds in the switch-internal network to pass traffic from the public internet to the corresponding internal components.

The name load-balancer comes from the fact that in the commercial version, when scaling out the system beyond just one pair of servers, the load-balancer instance becomes its own physical node and then handles multiple pairs of proxies behind it.

On the public interface, the load-balancer listens on port 5060 for UDP and TCP, as well as on 5061 for TLS connections. On the internal interface, it speaks SIP via UDP on port 5060 to the other system components, and listens for XMLRPC connections on TCP port 5060, which is used by the OSSBSS system to control the daemon.

Its config files reside in `/etc/ngcp-config/templates/etc/kamailio/lb/`, and changes to these files are applied by executing `ngcpcfg apply`.

Tip

The SIP load-balancer can be managed via the commands `/etc/init.d/kamailio-lb start`, `/etc/init.d/kamailio-lb stop` and `/etc/init.d/kamailio-lb restart`. Its status can be queried by executing `/etc/init.d/kamailio-lb status`. Also `ngcp-kamctl lb` and `ngcp-sercmd lb` are provided for querying kamailio functions, for example: `ngcp-sercmd lb htable.dump ipban`.

SIP Proxy/Registrar

The SIP proxy/registrar (or short *proxy*) is the work-horse of the sip:provider CE. It's also a separate Kamailio instance running in the switch-internal network and is connected to the provisioning database via MySQL, authenticates the endpoints, handles their registrations on the system and does the call routing based on the provisioning data. For each call, the proxy looks up the provisioned features of both the calling and the called party (either subscriber or domain features if it's a local caller and/or callee, or peering features if it's from/to an external endpoint) and acts accordingly, e.g. by checking if the call is blocked, by placing call-forwards if applicable and by normalizing numbers into the appropriate format, depending on the source and destination of a call.

It also writes start- and stop-records for each call, which are then transformed into call detail records (CDR) by the mediation system.

If the endpoints indicate negotiation of one or more media streams, the proxy also interacts with the *Media Relay* to open, change and close port pairs for relaying media streams over the sip:provider CE, which is especially important to traverse NAT.

The proxy listens on UDP port 5062 in the system-internal network. It cannot be reached directly from the outside, but only via the SIP load-balancer.

Its config files reside in `/etc/ngcp-config/templates/etc/kamailio/proxy/`, and changes to these files are applied by executing `ngcpcfg apply`.

Tip

The SIP proxy can be controlled via the commands `/etc/init.d/kamailio-proxy start`, `/etc/init.d/kamailio-proxy stop` and `/etc/init.d/kamailio-proxy restart`. Its status can be queried by executing `/etc/init.d/kamailio-proxy status`. Also `ngcp-kamctl proxy` and `ngcp-sercmd proxy` are provided for querying kamailio functions, for example: `ngcp-kamctl proxy ul show`.

SIP Back-to-Back User-Agent (B2BUA)

The SIP B2BUA (also called SBC within the system) decouples the first call-leg (calling party to sip:provider CE) from the second call-leg (sip:provider CE to the called party).

The software part used for this element is SEMS.

This element is typically optional in SIP systems, but it is always used for SIP calls (INVITE) that don't have the sip:provider CE as endpoint. It acts as application server for various scenarios (e.g. for feature provisioning via Vertical Service Codes and as Conferencing Server) and performs the B2BUA decoupling, topology hiding, caller information hiding, SIP header and Media

feature filtering, outbound registration, outbound authentication and call length limitation as well as Session Keep-Alive handler.

Due to the fact that typical SIP proxies (like the load-balancer and proxy in the sip:provider CE) do only interfere with the content of SIP messages where it's necessary for the SIP routing, but otherwise leave the message intact as received from the endpoints, whereas the B2BUA creates a new call leg with a new SIP message from scratch towards the called party, SIP message sizes are reduced significantly by the B2BUA. This helps to bring the message size under 1500 bytes (which is a typical default value for the MTU size) when it leaves the sip:provider CE. That way, chances of packet fragmentation are quite low, which reduces the risk of running into issues with low-cost SOHO routers at customer sides, which typically have problems with UDP packet fragmentation.

The SIP B2BUA only binds to the system-internal network and listens on UDP port 5080 for SIP messages from the load-balancer or the proxy, on UDP port 5040 for control messages from the cli tool and on TCP port 8090 for XMLRPC connections from the OSSBSS to control the daemon.

Its configuration files reside in `/etc/ngcp-config/templates/etc/sems`, and changes to these files are applied by executing `ngcpcfg apply`.

Tip

The SIP B2BUA can be controlled via the commands `/etc/init.d/sems start`, `/etc/init.d/sems stop` and `/etc/init.d/sems restart`. Its status can be queried by executing `/etc/init.d/sems status`

SIP App-Server

The SIP App-Server is an Asterisk instance used for voice applications like Voicemail and Reminder Calls. Asterisk uses the MySQL database as a message spool for voicemail, so it doesn't directly access the file system for user data. The voicemail plugin is a slightly patched version based on Asterisk 1.4 to make Asterisk aware of the sip:provider CE internal UUIDs for each subscriber. That way a SIP subscriber can have multiple E164 phone numbers, but all of them terminate in the same voicebox.

The App-Server listens on the internal interface on UDP port 5070 for SIP messages and by default uses media ports in the range from UDP port 10000 to 20000.

The configuration files reside in `/etc/ngcp-config/templates/etc/asterisk`, and changes to these files are applied by executing `ngcpcfg apply`.

Tip

The SIP App-Server can be controlled via the commands `/etc/init.d/asterisk start`, `/etc/init.d/asterisk stop` and `/etc/init.d/asterisk restart`. Its status can be queried by executing `/etc/init.d/asterisk status`

Media Relay

The Media Relay (also called *mediaproxy-ng* or *mediaproxy*) is a Kernel-based packet relay, which is controlled by the SIP proxy. For each media stream (e.g. a voice and/or video stream), it maintains a pair of ports in the range of port number 30000 to 40000. When the media streams are negotiated, mediaproxy opens the ports in user-space and starts relaying the packets to the addresses announced by the endpoints. If packets arrive from different source addresses than announced in the SDP body

of the SIP message (e.g. in case of NAT), the source address is implicitly changed to the address the packets are received from. Once the call is established and the mediaproxy has received media packets from both endpoints for this call, the media stream is pushed into the kernel and is then handled by a custom Sipwise iptables module to increase the throughput of the system and to reduce the latency of media packets.

The mediaproxy internally listens on UDP port 12222 for control messages from the SIP proxy. For each media stream, it opens two pairs of UDP ports on the public interface in the range of 30000 and 40000 per default, one pair on odd port numbers for the media data, and one pair on the next even port numbers for meta data, e.g. RTCP in case of RTP streams. Each endpoint communicates with one dedicated port per media stream (opposed to some implementations which use one pair for both endpoints) to avoid issues in determining where to send a packet to. The mediaproxy also sets the QoS/ToS/DSCP field of each IP packet it sends to a configured value, 184 (0xB8, *expedited forwarding*) by default.

The kernel-internal part of the mediaproxy is facilitated through an *iptables* module having the target name `MEDIAPROXY`. If any additional firewall or packet filtering rules are installed, it is imperative that this rule remains untouched and stays in place. Otherwise, if the rule is removed from iptables, the kernel will not be able to forward the media packets and forwarding will fall back to the user-space daemon. The packets will still be forwarded normally, but performance will be much worse under those circumstances, which will be especially noticeable when a lot of media streams are active concurrently. See the section on *Firewalling* for more information.

The mediaproxy configuration file is `/etc/ngcp-config/templates/etc/default/ngcp-mediaproxy-ng-dae mon`, and changes to this file are applied by executing `ngcpcfg apply`. The UDP port range can be configured via the `config.yml` file under the section `rtpproxy`. The QoS/ToS value can be changed via the key `qos.tos_rtp`.

Tip

The Media Relay can be controlled via the commands `/etc/init.d/ngcp-mediaproxy-ng-daemon start`, `/etc/init.d/ngcp-mediaproxy-ng-daemon stop` and `/etc/init.d/ngcp-mediaproxy-ng-daemon r estart`. Its status can be queried by executing `/etc/init.d/ngcp-mediaproxy-ng-daemon status`

2.1.2   Basic Call Flows

Endpoint Registration
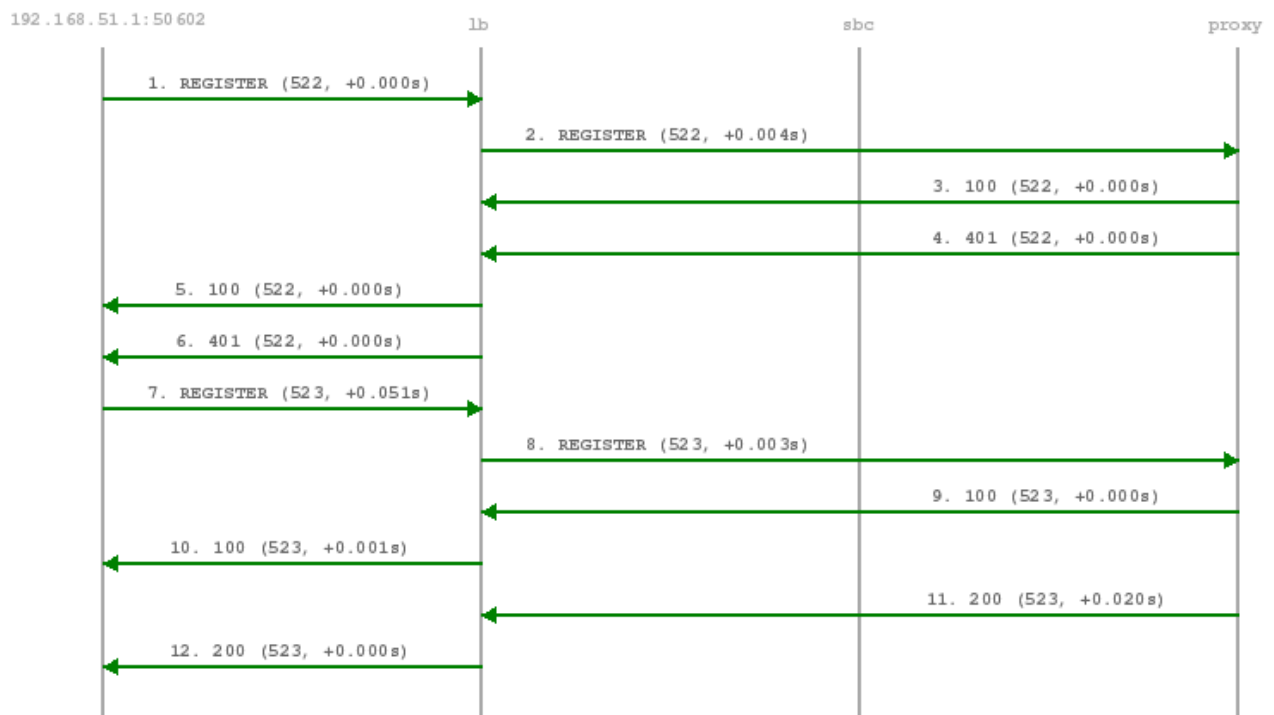
Figure 3: Registration Call-Flow

The subscriber endpoint starts sending a REGISTER request, which gets challenged by a 401. After calculating the response of the authentication challenge, it sends the REGISTER again, including the authentication response. The SIP proxy looks up the credentials of the subscriber in the database, does the same calculation, and if the result matches the one from the subscriber, the registration is granted.

The SIP proxy writes the content of the `Contact` header (e.g. `sip:me@1.2.3.4:1234;transport=UDP`) into its location table (in case of NAT the content is changed by the SIP load-balancer to the IP/port from where the request was received), so it knows where the reach a subscriber in case on an inbound call to this subscriber (e.g. `sip:someuser@example.org` is mapped to `sip:me@1.2.3.4:1234;transport=UDP` and sent out to this address).

If NAT is detected, the SIP proxy sends a OPTION message to the registered contact every 30 seconds, in order to keep the NAT binding on the NAT device open. Otherwise, for subsequent calls to this contact, the sip:provider PRO wouldn't be able to reach the endpoint behind NAT (NAT devices usually drop a UDP binding after not receiving any traffic for ~30-60 seconds).
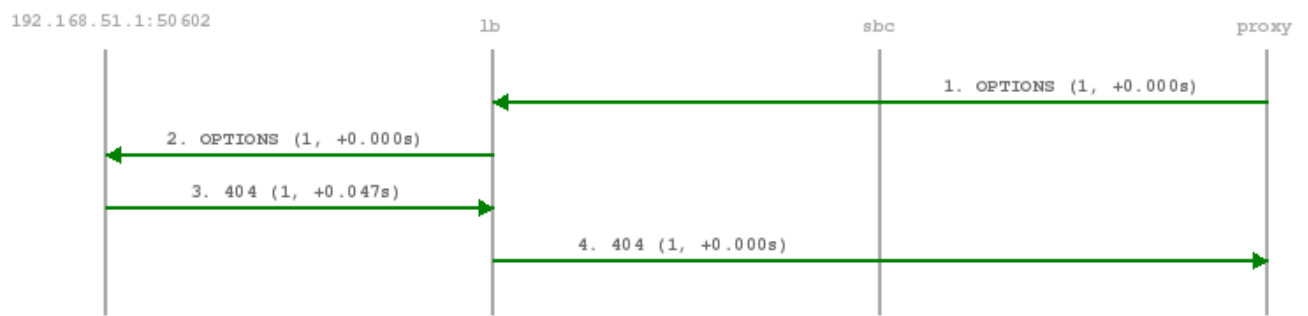
By default, a subscriber can register 5 contacts for an Address of Record (AoR, e.g. `sip:someuser@example.org`).

Basic Call
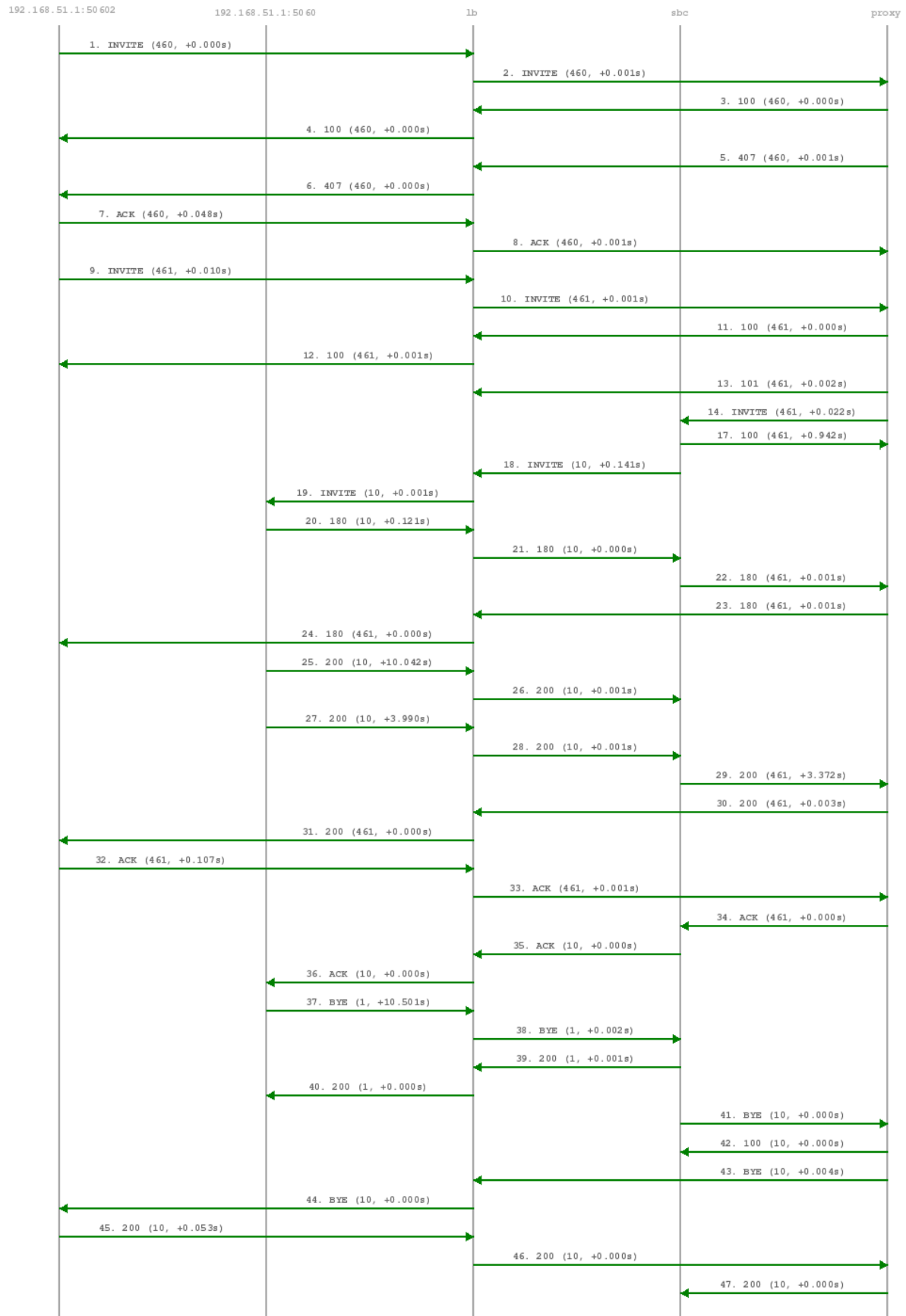
The calling party sends an INVITE (e.g. `sip:someuser@example.org`) via the SIP load-balancer to the SIP proxy. The proxy replies with an authorization challenge in the 407 response, and the calling party sends the INVITE again with authentication credentials. The SIP proxy checks if the called party is a local user. If it is, and if there is a registered contact found for this user, then (after various feature-related tasks for both the caller and the callee) the Request-URI is replaced by the URI of the registered contact (e.g. `sip:me@1.2.3.4:1234;transport=UDP`). If it's not a local user but a numeric user, a proper PSTN gateway is being selected by the SIP proxy, and the Request-URI is rewritten accordingly (e.g. `sip:+43123456789@2.3.4.5:5060`).

Once the proxy has finished working through the call features of both parties involved and has selected the final destination for the call, and - optionally - has invoked the Media Relay for this call, the INVITE is sent to the SIP B2BUA. The B2BUA creates a new INVITE message from scratch (using a new Call-ID and a new From-Tag), copies only various and explicitly allowed SIP headers from the old message to the new one, filters out unwanted media capabilities from the SDP body (e.g. to force audio calls to use G.711 as a codec) and then sends the new message via the SIP load-balancer to the called party.

SIP replies from the called party are passed through the elements back to the calling party (replacing various fields on the B2BUA to match the first call leg again). If a reply with an SDP body is received by the SIP proxy (e.g. a 183 or a 200), the Media Relay is invoked again to prepare the ports for the media stream.

Once the 200 is routed from the called party to the calling party, the media stream is fully negotiated, and the endpoints can start sending traffic to each outer (either end-to-end or via the Media Relay). Upon reception of the 200, the SIP proxy writes a start record for the accounting process. The 200 is also acknowledged with an ACK message from the calling party to the called party, according to the SIP 3-way handshake.

Either of the parties can tear down the media session at any time by sending a BYE, which is passed through to the other party. Once the BYE reaches the SIP proxy, it instructs the Media Relay to close the media ports, and it writes a stop record for accounting purposes. Both the start- and the stop-records are picked up by the *mediator* service in a regular interval and are converted into a Call Detail Record (CDR), which will be rated by the *rate-o-mat* process and can be billed to the calling party.

Session Keep-Alive

The SIP B2BUA acts as refresher for the Session-Timer mechanism as defined in RFC 4028. If the endpoints indicate support for the UPDATE method during call-setup, then the SIP B2BUA will use an UPDATE message if enabled per peer, domain or subscriber via Provisioning to check if the endpoints are still alive and responsive. Both endpoints can renegotiate the timer within a configurable range. All values can be tuned using the Admin Panel or the APIs using Peer-, Domain- and Subscriber-Preferences.

Tip

Keep in mind that the values being used in the signaling are always half the value being configured. So if you want to send a keep-alive every 300 seconds, you need to provision *sst_expires* to 600.

If one of the endpoints doesn't respond to the keep-alive messages or answers with `481 Call/Transaction Does Not Exist`, then the call is torn down on both sides. This mechanism prevents excessive over-billing of calls if one of the endpoints is not reachable anymore or "forgets" about the call. The BYE message sent by the B2BUA triggers a stop-record for accounting and also closes the media ports on the Media Relay to stop the call.

Beside the Session-Timer mechanism to prevent calls from being lost or kept open, there is a **maximum call length** of 21600 seconds per default defined in the B2BUA. This is a security/anti-fraud mechanism to prevent overly long calls causing excessive costs.

Voicebox Calls
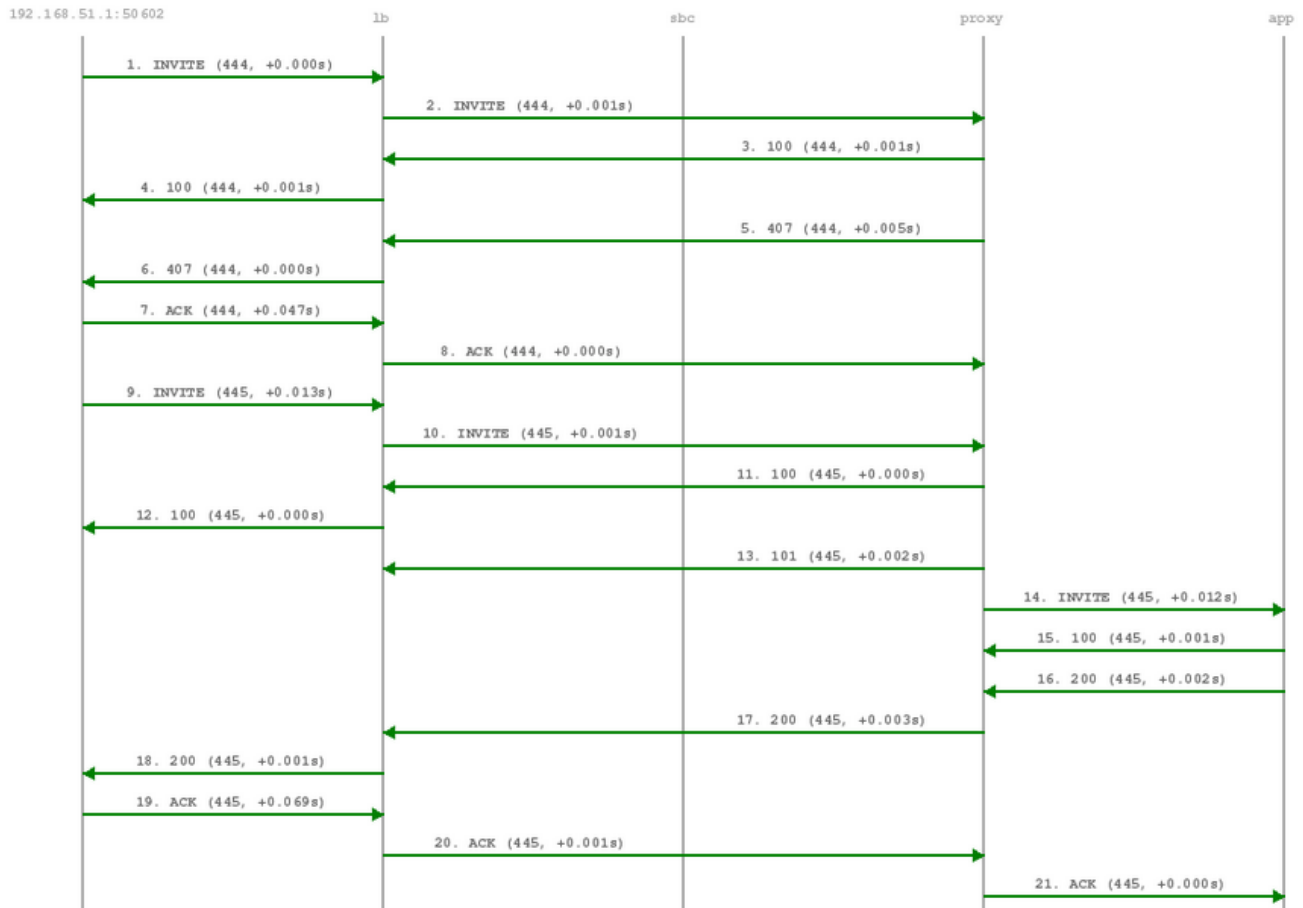
Calls to the Voicebox (both for callers leaving a voicemail message and for voicebox owners managing it via the IVR menu) are passed directly from the SIP proxy to the App-Server without a B2BUA. The App-Server maintains its own timers, so there is no risk of over-billing or overly long calls.

In such a case where an endpoint talks via the Media Relay to a system-internal endpoint, the Media Relay bridges the media streams between the public in the system-internal network.

In case of an endpoint leaving a new message on the voicebox, the Message-Waiting-Indication (MWI) mechanism triggers the sending of a unsolicited NOTIFY message, passing the number of new messages in the body. As soon as the voicebox owner dials into his voicebox (e.g. by calling `sip:voicebox@example.org` from his SIP account), another NOTIFY message is sent to his devices, resetting the number of new messages.

Important

The sip:provider CE does not require your device to subscribe to the MWI service by sending a SUBSCRIBE (it would rather reject it). On the other hand, the endpoints need to accept unsolicited NOTIFY messages (that is, a NOTIFY without a valid subscription), otherwise the MWI service will not work with these endpoints.

3   Upgrading from previous versions

3.1   Upgrade from previous versions to mr3.5.1

The sip:provider CE system upgrade to mr3.5.1 will be performed in a couple of tasks:

1. Upgrade NGCP software packages

2. Upgrade NGCP configuration templates

3. Upgrade base system within Debian (v7) to latest package versions

For upgrading the sip:provider CE to release mr3.5.1, execute the following commands:

```
NGCP_CURRENT_VERSION=$(cat /etc/ngcp_version)
sed -i "s/$NGCP_CURRENT_VERSION/mr3.5.1/" /etc/apt/sources.list.d/sipwise.list
apt-get update
apt-get install ngcp-upgrade-mr3.5.1-ce
```

Note

sip:provider CE can be upgraded to mr3.5.1 from previous release or previous build only, otherwise you will get an error *"Unable to locate package ngcp-upgrade-mr3.5.1-ce"* (in this case, you need to upgrade sip:provider CE to previous release first).

Run the upgrade script as *root* like this:

```
ngcp-upgrade
```

The upgrade script will ask you to confirm that you want to start. Read the given information **carefully**, and if you agree, proceed with *y*.

The upgrade process will take several minutes, depending on your network connection and server performance. After everything has been updated successfully, it will finally ask you to reboot your system. Confirm to let the system reboot (it will boot with an updated kernel).

Once up again, double-check your config file */etc/ngcp-config/config.yml* (sections will be rearranged now and will contain more parameters) and your domain/subscriber/peer configuration and test the setup. You can find a backup of some important configuration files of your existing installation under */var/backup/ngcp-mr3.5.1-** (where *** is a place holder for a timestamp) in case you need to roll back something at any time. A log file of the upgrade procedure is available at */var/backup/ngcp-mr3.5.1-*/upgrade.log*.

Note

sip:provider CE mr3.5.1 requires at least 2GB of RAM available as the minimum requirements of the installation section, otherwise certain features won't work and you will run into arbitrary issues.

---

4 Initial Installation

4.1 Prerequisites

For an initial installation of the sip:provider CE, it is mandatory that your production environment meets the following criteria:

HARDWARE REQUIREMENTS

- Recommended: Dual-core, x86_64 compatible, 3GHz, 4GB RAM, 128GB HDD

- Minimum: Single-core, x86_64 compatible, 1GHz, 2GB RAM, 16GB HDD

SUPPORTED OPERATING SYSTEMS

- Debian Wheezy (v7) 64-bit

INTERNET CONNECTION

- Hardware needs connection to the Internet

⚠ **Important**

Only **Debian Wheezy (v7) 64-bit** is currently supported as a host system for the sip:provider CE.

⚠ **Important**

It is **HIGHLY** recommended that you use a **dedicated server** (either a physical or a virtual one) for sip:provider CE, because the installation process will wipe out existing MySQL databases and modify several system configurations.

4.2 Using the NGCP install CD (recommended)

The custom Sipwise NGCP install CD provides the ability to easily install sip:provider CE, including automatic partitioning and installation of the underlying Debian system. You can install the latest available or Long Term Support (LTS) sip:provider CE release using sip:provider CE install CD image (checksums: sha1, md5).

⚠ **Important**

The NGCP install CD automatically takes care of partitioning, any present data will be overwritten! While the installer prompts for the disk that should be used for installation before its actual execution, it's strongly recommended to boot the ISO in an environment with empty disks or disks that you don't plan to use for anything else than the newly installed NGCP system.

To configure network options please choose the according boot menu entries with either `DHCP` or `static NW` (static network configuration) in its name. Press the `<tab>` key on the menu entry you want to boot, then adjust the `ip=...` and `dns=...` boot options as needed.

Tip

When DHCP is available in your infrastructure then you shouldn't have to configure anything, just choose `DHCP` and press enter. If network configuration still doesn't work as needed a console based network configuration system will assist you in setting up your network configuration.

Also, you can use sip:provider CE install CD to boot the Grml (Debian based live system) rescue system, check RAM using a memory testing tool or install plain Debian squeeze/wheezy system for manual installation using NGCP installer.

4.3   Using the NGCP installer

4.3.1   Installing the Operating System

You need to install Debian Wheezy (v7) 64-bit on the server. A **basic** installation without any additional task selection (like *Desktop System*, *Web Server* etc.) is sufficient.

Tip

Sipwise recommends using the latest Netinstall ISO as installation medium.

Important

If you use other kinds of installation media (e.g. provided by your hosting provider), prepare for some issues that might come up during installation. For example, you might be forced to manually resolve package dependencies in order to install the sip:provider CE. Therefore, it is HIGHLY RECOMMENDED to use a clean Debian installation to simplify the installation process.

Note

If you installed your system using the Debian CDs/DVDs (so neither using the NGCP install CD nor the Debian Netinstall ISO) apt-get might prompt to insert disk to proceed during NGCP installation. The prompt won't be visible for you and installation hangs. Please disable the cdrom entries in /etc/apt/sources.list and enable a Debian mirror (e.g. http://http.debian.net/debian) instead.

Using special Debian setups

If you plan to install the sip:provider CE on Virtual Hosting Providers like *Dreamhost* with their provided Debian installer, you might need to manually prepare the system for the NGCP installation, otherwise the installer will fail installing certain package versions required to function properly.

Using Dreamhost Virtual Private Server

A Dreamhost virtual server uses apt-pinning and installs specific versions of MySQL and apache, so you need to clean this up beforehand.

```
apt-get remove --purge mysql-common ndn-apache22
mv /etc/apt/preferences /etc/apt/preferences.bak
apt-get update
apt-get dist-upgrade
```

> **⚠ Warning**
>
> Be aware that this step will break your web-based system administration provided by Dreamhost. Only do it if you are certain that you won't need it.

4.3.2 Installing the sip:provider CE

The sip:provider CE is based on the *Sipwise NGCP*, so download and install the latest *Sipwise NGCP* installer package:

```
PKG=ngcp-installer-latest.deb
wget http://deb.sipwise.com/spce/${PKG}
dpkg -i ${PKG}
```

Run the installer as root user:

```
ngcp-installer
```

> **Note**
>
> You can find the previous versions of *Sipwise NGCP* installer package here.

The installer will ask you to confirm that you want to start the installation. Read the given information **carefully**, and if you agree, proceed with *y*.

The installation process will take several minutes, depending on your network connection and server performance. If everything goes well, the installer will (depending on the language you use), show something like this:

```
Installation finished. Thanks for choosing NGCP sip:provider Community Edition.
```

During the installation, you can watch the background processing by executing the following command on a separate console:

```
tail -f /tmp/ngcp-installer.log
```

4.4 Using a pre-installed virtual machine

For quick test deployments, pre-installed virtualization images are provided. These images are intended to be used for quick test, not recommended for production use.

4.4.1 Vagrant box for VirtualBox

Vagrant is an open-source software for creating and configuring virtual development environments. Sipwise provides a so called Vagrant base box for your service, to easily get direct access to your own sip:provider CE Virtual Machine without any hassles.

Note

The following software must be installed to use Vagrant boxes:

- VirtualBox v.4.3.16+

- Vagrant v.1.6.5+

Get your copy of sip:provider CE by running:

```
vagrant init spce-mr3.5.1 http://deb.sipwise.com/spce/images/sip_provider_CE_mr3.5.1 ←
    _vagrant.box
vagrant up
```

As soon as the machine is up and ready you should have your local copy of sip:provider CE with the following benefits:

- all the software and database are automatically updated to the latest available version

- the system is configured to use your LAN IP address (received over DHCP)

- basic SIP credentials to make SIP-2-SIP calls out of the box are available

Use the following command to access the terminal:

```
vagrant ssh
```

or login to Administrator web-interface at https://127.0.0.1:1443/login/admin (with default user *administrator* and password *administrator*).

There are two ways to access VM resources, through NAT or Bridge interface:

Note

a.b.c.d is IP address of VM machine received from DHCP; x.y.z.p is IP address of your host machine

---

Table 1: Vagrant based VirtualBox VM interfaces:

Description	Host-only address	LAN address	Notes
SSH	ssh://127.0.0.1:2222	ssh://a.b.c.d:22 or ssh://x.y.z.p:2222	Also available via "*vagrant ssh*"
Administrator interface	https://127.0.0.1:1443/-login/admin	https://a.b.c.d:1443/login/-admin or https://x.y.z.p:1443/login/-admin	

Table 1: (continued)

Description	Host-only address	LAN address	Notes
New Customer self care interface	https://127.0.0.1:1443	https://a.b.c.d:1443 or https://x.y.z.p:1443	new self-care interface based on powerful ngcp-panel framework
Old Customer self care interface	https://127.0.0.1:22443	https://a.b.c.d:443 or https://x.y.z.p:22443	will be removed in upcoming releases
Provisioning interfaces	https://127.0.0.1:2443	https://a.b.c.d:2443 or https://x.y.z.p:2443	
SIP interface	not available	sip://a.b.c.d:5060	Both TCP and UDP are available.

Note

VM ports smaller then 1024 mapped to ports 22<vm_port> through NAT, otherwise root on host machine requires to map them. It means SSH port 22 mapped to port 2222, WEB port 443 → 22443.

VM IP address (a.b.c.d), as well as SIP credentials will be printed to terminal during "vagrant up" stage, e.g.:

```
[20_add_sip_account] Adding SIP credentials...
[20_add_sip_account]   - removing domain 192.168.1.103 with subscribers
[20_add_sip_account]   - adding domain 192.168.1.103
[20_add_sip_account]   - adding subscriber 43991002@192.168.1.103 (pass: 43991002)
[20_add_sip_account]   - adding subscriber 43991003@192.168.1.103 (pass: 43991003)
[20_add_sip_account]   - adding subscriber 43991004@192.168.1.103 (pass: 43991004)
[20_add_sip_account]   - adding subscriber 43991005@192.168.1.103 (pass: 43991005)
[20_add_sip_account]   - adding subscriber 43991006@192.168.1.103 (pass: 43991006)
[20_add_sip_account]   - adding subscriber 43991007@192.168.1.103 (pass: 43991007)
[20_add_sip_account]   - adding subscriber 43991008@192.168.1.103 (pass: 43991008)
[20_add_sip_account]   - adding subscriber 43991009@192.168.1.103 (pass: 43991009)
[20_add_sip_account] You can USE your VM right NOW:  https://192.168.1.103:1443/login/admin
```

To turn off your sip:provider CE virtual machine, just type:

```
vagrant halt
```

To completely remove sip:provider CE virtual machine, use:

```
vagrant destroy
vagrant box remove spce-mr3.5.1
```

Further documentation for Vagrant is available at the official Vagrant website.

Vagrant usage tips:

- Default SSH login is *root* and password is *sipwise*. SSH connection details can be displayed via:

```
vagrant ssh-config
```

- You can download a Vagrant box for VirtualBox from here manually (checksums: sha1, md5).

4.4.2 VirtualBox image

You can download a VirtualBox image from here (checksums: sha1, md5). Once you have downloaded the file you can import it to VirtualBox via its import utility.

The format of the image is *ova*. If you have VirtualBox 3.x running, which is not compatible with *ova* format, you need to extract the file with any *tar* compatible software and import the *ovf* file which is inside the archive.

On Linux, you can do it like this:

```
tar xvf sip_provider_CE_mr3.5.1_virtualbox.ova
```

On Windows, right-click on the ova file, choose *Open with* and select *WinZIP* or *WinRAR* or any other application able to extract *tar* archives. Extract the files to any place and import the resulting *ovf* file in VirtualBox.

Considerations when using this virtual machine:

- You will need a 64bit guest capable VirtualBox setup.

- The root password is *sipwise*

- You should use *bridge mode* networking (adjust your bridging interface in the virtual machine configuration) to avoid having the sip:provider CE behind NAT.

- You'll need to adjust your timezone and keyboard layout.

- The network configuration is set to DHCP. You'll need to change it to the appropriate static configuration.

- As the virtual image is a static file, it won't contain the most updated versions of our software. Please upgrade the system via apt as soon as you boot it for the first time.

4.4.3 VMware image

You can download a VMware image from here (checksums: sha1, md5). Once you have downloaded the file just extract the *zip* file and copy its content to your virtual machines folder.

Considerations when using this virtual machine:

- You will need a 64bit guest capable vmware setup.

- The root password is *sipwise*

- You'll need to adjust your timezone and keyboard layout.

- The network configuration is set to DHCP. You'll need to change it to the appropriate static configuration.

- As the virtual image is a static file, it won't contain the most updated versions of our software. Please upgrade the system via apt as soon as you boot it for the first time.

5 Initial System Configuration

After the installation went through successfully, you are ready to adapt the system parameters to your needs to make the system work properly.

5.1 Network Configuration

If you have only one network card inside your system, its device name is *eth0*, it's configured and only IPV4 is important to you then there should be nothing to do for you at this stage. If multiple network cards are present, your network card does *not* use *eth0* for its device name or you need IPv6 then the only parameter you need to change at this moment is the listening address for your SIP services.

To do this, you have to specify the interface where your listening address is configured, which you can do with the following command (assuming your public interface is *eth0*):

```
ngcp-network --set-interface=eth0 --ip=auto --netmask=auto
ngcp-network --move-from=lo --move-to=eth0 --type=web_ext --type=sip_ext --type=rtp_ext -- ←↩
    type=ssh_ext
```

If you want to enable IPv6 as well, you have to set the address on the proper interface as well, like this (assuming you have an IPv6 address *fdda:5cc1:23:4:0:0:0:1f* on interface *eth0*):

```
ngcp-network --set-interface=eth0 --ipv6='fdda:5cc1:23:4:0:0:0:1f'
```

Tip

Always use a full IPv6 address with 8 octets, leaving out zero octets (e.g. `fdda:5cc1:23:4::1f`) is **not allowed**.

If you haven't fully configured your network interfaces, do this by adapting also the file */etc/network/interfaces*:

```
vim /etc/network/interfaces
```

Add or adapt your interface configuration accordingly. For example, if you just want to use the system in your internal network 192.168.0.0/24, it could look something like this:

```
auto lo
iface lo inet loopback

auto eth0
```

```
iface eth0 inet static
        address 1.2.3.4
        netmask 255.255.255.0
        gateway 1.2.3.1
        dns-nameservers 8.8.8.8
        dns-search yourdomain.com
```

```
/etc/init.d/networking restart
```

5.2 Apply Configuration Changes

In order to apply the changes you made to */etc/ngcp-config/config.yml*, you need to execute the following command to re-generate your configuration files and to automatically restart the services:

```
ngcpcfg apply
```

Tip
At this point, your system is ready to serve.

5.3 Start Securing Your Server

During installation, the system user *cdrexport* is created. This jailed system account is supposed to be used to export CDR files via sftp/scp. Set a password for this user by executing the following command:

```
passwd cdrexport
```

The installer has set up a MySQL database on your server. You need to set a password for the MySQL root user to protect it from unauthorized access by executing this command:

```
mysqladmin password <your mysql root password>
```

For the *Administrative Web Panel* located at *https://<your-server-ip>:1443/login/admin*, a default user *administrator* with password *administrator* has been created. Connect to the panel (accept the SSL certificate for now) using those credentials and change the password of this user by going to *Settings→Administrators* and click the *Edit* when hovering over the row.

5.4 Configuring the Email Server

The NGCP installer will install *mailx* (which has *Exim4* as MTA as a default dependency) on the system, however the MTA is not configured by the installer. If you want to use the *Voicemail-to-Email* feature of the Voicebox, you need to configure your MTA properly. If you are fine to use the default MTA *Exim4*, execute the following command:

```
dpkg-reconfigure exim4-config
```

Depending on your mail setup in your environment (whether to use a smarthost or not), configure Exim accordingly. In the most simple setup, apply the following options when prompted for it:

- **General type of mail configuration:** `internet site;mail is sent and received directly using SMTP`

- **System mail name:** the FQDN of your server, e.g. `ce.yourdomain.com`

- **IP-addresses to listen on for incoming SMTP connections:** `127.0.0.1`

- **Other destinations for which mail is accepted:** the FQDN of your server, e.g. `ce.yourdomain.com`

- **Domains to relay mail for:** leave empty

- **Machines to relay mail for:** leave empty

- **Keep number of DNS-queries minimal (Dial-on-Demand)?** `No`

- **Delivery method for local mail:** `mbox format in /var/mail/`

- **Split configuration into small files?** `No`

⚠ **Important**

You are free to install and configure any other MTA (e.g. postfix) on the system, if you are more comfortable with that.

5.5 Advanced Network Configuration

You have a typical test deployment now and you are good to go, however you may need to do extra configuration depending on the devices you are using and functionality you want to achieve.

5.5.1 Audiocodes devices workaround

As reported by many users, Audiocodes devices suffer from a problem where they replace `127.0.0.1` address in Record-Route headers (added by the sip:provider CE's internal components) with its own IP address. The problem has been reported to Audiocodes but as of end 2012 the fixed firmware is not available yet so supposedly the whole range of Audiocodes devices, including but not limited to the MP202, MP252 CPEs as well as Audiocodes media gateways, is malfunctioning. As a workaround, you may change the internal IP address from `127.0.0.1` to some dummy network interface. Please execute the following command (in this example `192.168.2.2` is a new internal IP address):

```
ifconfig dummy0 192.168.2.2 netmask 255.255.255.0
```

Adapt your */etc/network/interfaces* file accordingly:

```
auto dummy0
iface dummy0 inet static
address 192.168.2.2
netmask 255.255.255.0
```

Update the network configuration in the sip:provider CE:

```
ngcp-network --set-interface=dummy0 --ip=auto --netmask=auto
ngcp-network --move-from=lo --move-to=dummy0 --type=sip_int --type=web_int
```

Refer to the Network Configuration chapter for more details about the `ngcp-network` tool.

Apply configuration:

```
ngcpcfg apply
```

5.5.2 Extra SIP listening ports

By default, the load-balancer in sip:provider CE listens on the UDP and TCP ports 5060 (*kamailio→lb→port*) and TLS port 5061 (*kamailio→lb→tls→port*). Should you need to setup one or more extra non-standard listening ports in addition to those standard ports, please edit the *kamailio→lb→extra_sockets* option in your */etc/ngcp-config/config.yml* file.

The correct format consists of a label and value like this:

```
    extra_sockets:
      port_5064: udp:10.15.20.108:5064
      test: udp:10.15.20.108:6060
```

The label is shown in `outbound_socket` peer preference (if you want to route calls to specific peer out via specific socket); the value must contain a transport specification as in example above (udp, tcp or tls).

> **! Important**
>
> The media relay uses one main primary external IP address. You should make sure it is reachable to all of your subscribers and peers (or disable the media relay for subscribers by checking the `never_use_rtpproxy` preference if they have routable IP addresses) - refer to the Security and Maintenance chapter for more details on firewalling.

Apply configuration:

```
ngcpcfg apply
```

5.6 What's next?

To test and use your installation, you need to follow these steps now:

1. Create a SIP domain

2. Create some SIP subscribers

3. Register SIP endpoints to the system

4. Make local calls and test subscriber features

5. Establish a SIP peering to make PSTN calls

Please read the next chapter for instructions on how to do this.

6 Administrative Configuration

To be able to configure your first test clients, you will need a Customer, a SIP domain and some subscribers in this domain. Throughout this steps, let's assume you're running the NGCP on the IP address *1.2.3.4*, and you want this IP to be used as SIP domain. This means that your subscribers will have an URI like *user1@1.2.3.4*.

Tip

You can of course set up a DNS name for your IP address (e.g. letting sip.yourdomain.com point to 1.2.3.4) and use this DNS name throughout the next steps, but we'll keep it simple and stick directly with the IP as a SIP domain for now.

Warning

Once you started adding subscribers to a SIP domain, and later decide to change the domain, e.g. from 1.2.3.4 to sip.yourdomain.com, you'll need to recreate all your subscribers in this new domain. It's currently not possible to easily change the domain part of a subscriber.

Go to the *Administrative Web Panel* (*Admin Panel*) running on *https://<ce-ip>:1443/login/admin* and follow the steps below. The default user on the system is *administrator* with the password *administrator*, if you haven't changed it already in [?simpara].
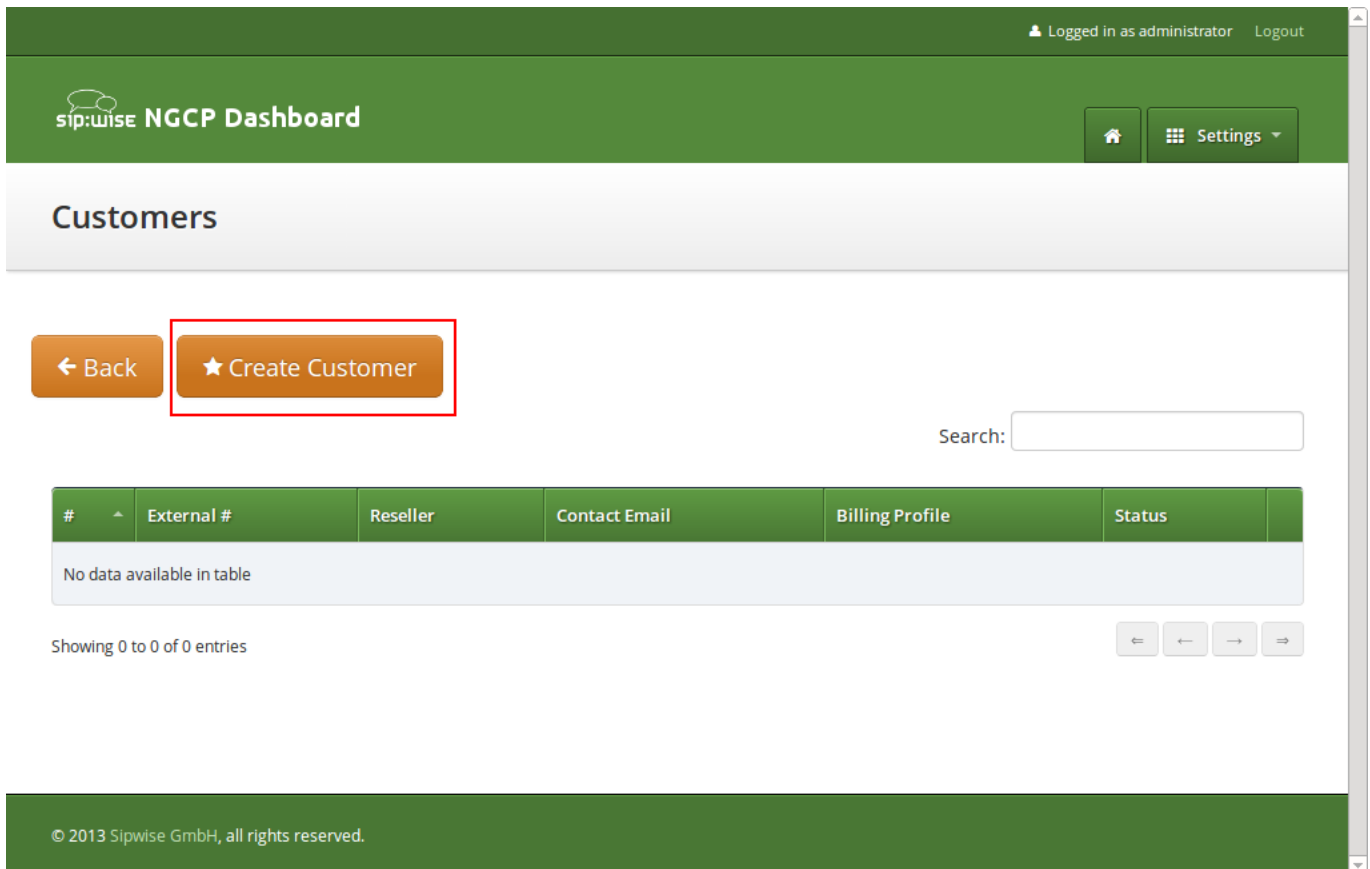
6.1 Creating a Customer

A Customer is a special type of contract on the system acting as billing container for SIP subscribers. You can create as many SIP subscribers within a Customer as you want.

To create a Customer, got to *Settings→Customers*.
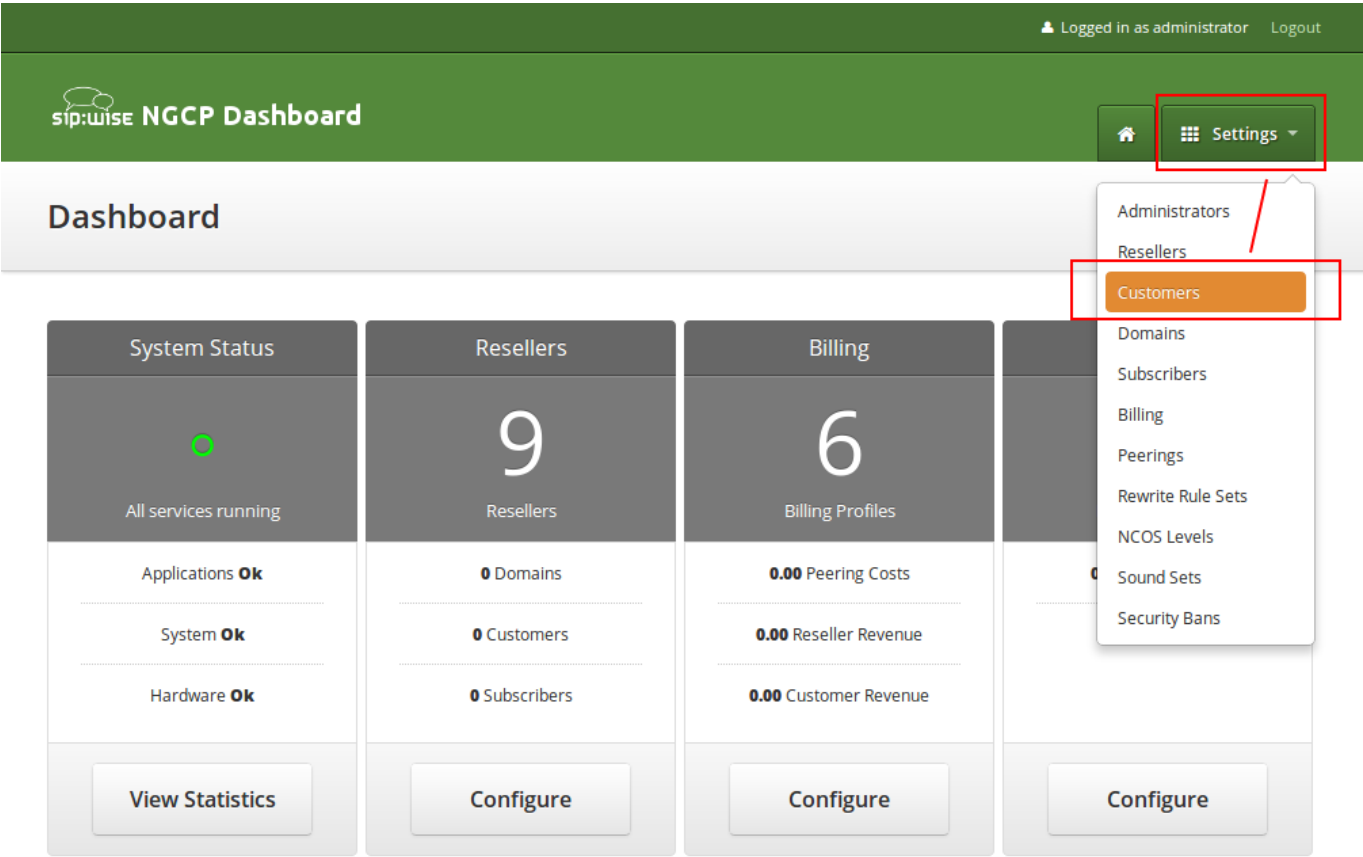
Click on *Create Customer*.

Each *Customer* needs a *Contact*. We can either reuse the default one, but for a clean setup, we create a new *Contact* for each *Customer* to be able to identify the *Customer*. Click on *Create Contact* to create a new *Contact*.

We assign the Contact to the default *Reseller*. You can create a new one if you want, but for a simple setup the default *Reseller* is sufficient. Select the *Reseller* and enter the contact details (at least an *Email* is required), then press *Save*.

You will be redirected back to the *Contract* form. The newly created *Contact* is selected by default now, so you only have to select a *Billing Profile*. Again you can create a new one on the fly, but we will go with the default profile for now. Select it and press *Save*.

You will now see your first *Customer* in the list. Hover over the customer and click *Details* to view the details.

6.2 Creating a Subscriber

In your *Customer* details view, click on the *Subscribers* row, then click the *Create Subscriber*.

As you can see, we don't have any *SIP Domains* yet, so click on *Create Domain* to create one.

Select the *Reseller* (make sure to use the same reseller where your *Customer* is created in) and enter your domain name, then press *Save*.

Your *Domain* will be preselected now, so fill out the rest of the form:

- **Web Username**: This is the user part of the username the subscriber may use to log into her *Customer Self Care Interface*. The user part will be automatically suffixed by the SIP domain you choose for the **SIP URI**. Usually the web username is identical to the **SIP URI**, but you may choose a different naming schema.

⚠ **Caution**

The web username needs to be unique. The system will return a fault if you try to use the same web username twice.

- **Web Password**: This is the password for the subscriber to log into her *Customer Self Care Interface*. It must be at least 6 characters long.

- **E164 Number**: This is the telephone number mapped to the subscriber, separated into *Country Code (CC)*, *Area Code (AC)* and *Subscriber Number (SN)*. For the first tests, you can set a made-up number here and change it later when you get number blocks assigned by your PSTN interconnect partner. So in our example, we'll use *43* as CC, *99* as AC and *1001* as SN to form the phantasy number *+43 99 1001*.

Tip

This number can actually be used to place calls between local subscribers, even if you don't have any PSTN interconnection. This comes in handy if you use phones instead of soft-clients for your tests. The format in which this number can be dialled so the subscriber is reached is defined in Section 6.6.

> **Important**
>
> NGCP allows single subscriber to have multiple E.164 numbers to be used as aliases for receiving incoming calls. Also NGCP supports "implicit" extensions, e.g. if a subscriber has number 012345, but somebody calls 012345100, then it first tries to send the call to number 012345100 (even though the user is registered as myusername), and only after 404 it falls back to the user-part for which the user is registered.

- **SIP Username**: The user part of the SIP URI for your subscriber.

- **SIP Domain**: The domain part of the SIP URI for your subscriber.

- **SIP Password**: The password of your subscriber to authenticate on the SIP proxy. It must be at least 6 characters long.

- **Status**: You can lock a subscriber here, but for creating one, you will most certainly want to use *active*.

- **External ID**: You can provision an arbitrary string here (e.g. an ID of a 3rd party provisioning/billing system).

- **Administrative**: If you have multiple subscribers in one account and set this option for one of them, this subscriber can administrate other subscribers via the *Customer Self Care Interface*.

Repeat the creation of *Customers* and *Subscribers* for all your test accounts. You should have at least 3 subscribers to test all the functionality of the NGCP.
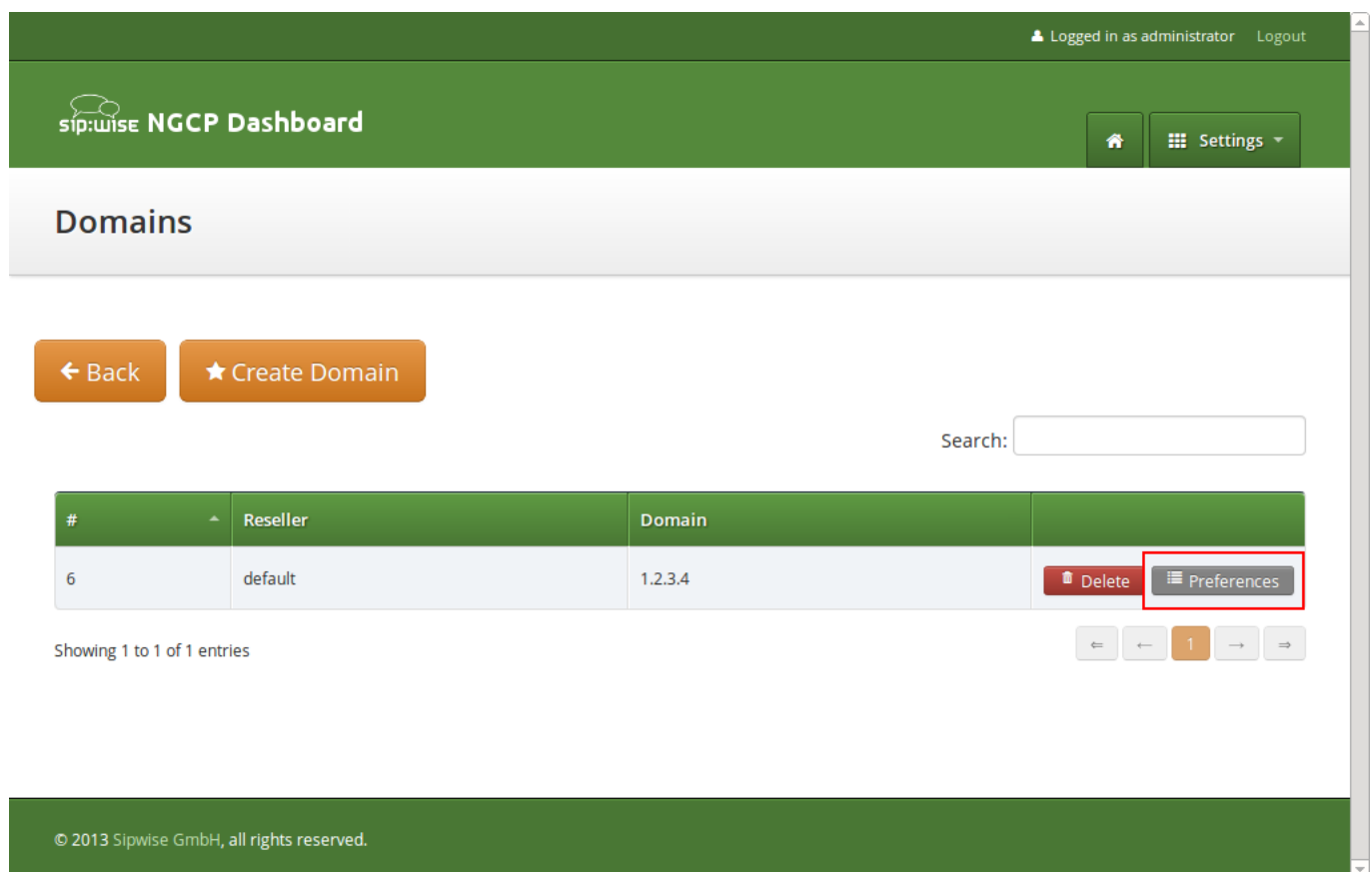
> **Tip**
>
> At this point, you're able to register your subscribers to the NGCP and place calls between these subscribers.

You should now revise the *Domain* and *Subscriber* Preferences.

6.3   Domain Preferences

The *Domain Preferences* are the default settings for *Subscriber Preferences*, so you should set proper values there if you don't want to configure each subscriber separately. You can later override these settings in the *Subscriber Preferences* if particular subscribers need special settings.

To configure your *Domain Preferences*, go to *Settings→Domains* and click on the *Preferences* button of the domain you want to configure.

The most important settings are in the group *Number Manipulations*, where you can configure where from a SIP message to take numbers from for incoming messages, where in the SIP messages to put which numbers for outgoing SIP messages, and how these numbers are normalized to E164 format and vice versa.

To assign a *Rewrite Rule Set* to a *Domain*, create a set first as described in Section 6.6, then assign it to the domain by editing the *rewrite_rule_set* preference.

Select the *Rewrite Rule Set* and press *Save*.

Then, select the field you want the *User Provided Number* to be taken from for inbound INVITE messages. Usually the *From-Username* should be fine, but you can also take it from the *Display-Name* of the From-Header, and other options are available as well.

6.4 Subscriber Preferences

You can override the *Domain Preferences* on a subscriber basis as well. Also, there are *Subscriber Preferences* which don't have a default value in the *Domain Preferences*.

To configure your *Subscriber*, go to *Settings→Subscribers* and click *Details* on the row of your subscriber. There, click on the *Preferences* button on top.

You want to look into the *Number Manipulations* and *Access Restrictions* options in particular, which control what is used as user-provided and network-provided calling numbers.

- For outgoing calls, you may define multiple numbers or patterns to control what a subscriber is allowed to send as user-provided calling numbers using the *allowed_clis* preference.

- If *allowed_clis* does not match the number sent by the subscriber, then the number configured in *cli* (the network-provided number) preference will be used as user-provided calling number also.

- You can override any user-provided number coming from the subscriber using the *user_cli* preference.

6.5   Creating Peerings

If you want to terminate calls at or allow calls from 3rd party systems (e.g. PSTN gateways, SIP trunks), you need to create SIP peerings for that. To do so, go to *Settings→Peerings*. There you can add peering groups, and for each peering group add peering servers and rules controlling which calls are routed over these groups. Every peering group needs a peering contract for correct interconnection billing.

6.5.1   Creating Peering Groups

Click on *Create Peering Group* to create a new group.

In order to create a group, you must select a peering contract. You will most likely want to create one contract per peering group.

Click on *Create Contract* create a *Contact*, then select a *Billing Profile*.

Click *Save* on the *Contacts* form, and you will get redirected back to the form for creating the actual *Peering Group*. Put a name, priority and description there, for example:

- **Peering Contract:** select the id of the contract created before

- **Name:** test group

- **Priority:** 1

- **Description:** peering to a test carrier

The *Priority* option defines which *Peering Group* to favor if two peering groups have peering rules matching an outbound call. *Peering Rules* are described below.

Then click *Save* to create the group.

6.5.2 Creating Peering Servers

In the group created before, you need to add peering servers to route calls to and receive calls from. To do so, click on *Details* on the row of your new group in your peering group list.

To add your first *Peering Server*, click on the *Create Peering Server* button.

In this example, we will create a peering server with IP *2.3.4.5* and port *5060*:

- **Name**: `test-gw-1`

- **IP Address:** `2.3.4.5`

- **Hostname:** leave empty

- **Port:** `5060`

- **Protocol:** `UDP`

- **Weight:** `1`

- **Via Route:** `None`

Click *Save* to create the peering server.

Tip

The *hostname* field for a peering server is optional. Usually, the IP address of the peer is used as domain part in the Request URI. Some peers may require you to set a particular hostname instead of the IP address there, which can be done by filling in this field. The IP address must always be given though, and the request will always be sent to the IP address, no matter what you put into the *hostname* field.

Tip

If you want to add a peering server with an IPv6 address, enter the address without surrounding square brackets into the *IP Address* column, e.g. *::1*.

You can force an additional hop (e.g. via an external SBC) towards the peering server by using the *Via Route* option. The available options you can select there are defined in `/etc/ngcp-config/config.yml`, where you can add an array of SIP URIs in `kamailio→lb→external_sbc` like this:

```
kamailio:
  lb:
    external_sbc:
```

```
    - sip:192.168.0.1:5060
    - sip:192.168.0.2:5060
```

Execute `ngcpcfg apply`, then edit your peering server and select the hop from the *Via Route* selection.

Once a peering server has been created, this server can already send calls to the system.

⚠ Important

To be able to send outbound calls towards the servers in the *Peering Group*, you also need to define *Peering Rules*. They specify which source and destination numbers are going to be terminated over this group. To create a rule, click the *Create Peering Rule* button.

Since the previously created peering group will be the only one in our example, we have to add a default rule to route *all* calls via this group. To do so, create a new peering rule with the following values:

- **Callee Prefix:** leave empty

- **Callee Pattern:** leave empty

- **Caller Pattern:** leave empty

- **Description:** `Default Rule`

Then click *Save* to add the rule to your group.

Tip

If you set the caller or callee rules to refine what is routed via this peer, enter all phone numbers in full E.164 format, that is *<cc><ac><sn>*. TIP: The *Caller Pattern* field covers the whole URI including the subscriber domain, so you can only allow certain domains over this peer by putting for example `@example\.com` into this field.

Peering Servers

← Back   ★ Create Peering Server

Search: [   ]

# ▲	Name	IP Address	Hostname	Port	Protocol	Weight	Via Route Set	
3	test-gw-1	2.3.4.5		5060	1	1		

Showing 1 to 1 of 1 entries   ⇐ ← **1** → ⇒

Peering Rules

★ Create Peering Rule

Peering rule successfully created

Search: [   ]

# ▲	Callee Prefix	Callee Pattern	Caller Pattern	Description	
1				Default Rule	

Showing 1 to 1 of 1 entries   ⇐ ← **1** → ⇒

Important

⚠

The selection of peering servers for outbound calls is done in the following order: **1.** whether caller or callee pattern matched. **2.** length of the callee prefix. **3.** priority of the peering group. **4.** weight of the peering servers in the selected peering group. After one or more peering group(s) is matched for an outbound call, all servers in this group are tried, according to their weight (lower weight has more precedence). If a peering server replies with SIP codes `408`, `500` or `503`, or if a peering server doesn't respond at all, the next peering server in the current peering group is used as a fallback, one after the other until the call succeeds. If no more servers are left in the current peering group, the next group which matches the peering rules is going to be used.

6.5.3   Authenticating and Registering against Peering Servers

Proxy-Authentication for outbound calls

If a peering server requires the SPCE to authenticate for outbound calls (by sending a 407 as response to an INVITE), then you have to configure the authentication details in the *Preferences* view of your peer host.

Peering Servers

[← Back]   [★ Create Peering Server]

Search: [   ]

# ▲	Name	IP Address	Hostname	Port	Protocol	Weight			
1	test-gw-1	2.3.4.5		5060	1	1	☑ Edit	🗑 Delete	☰ Preferences

Showing 1 to 1 of 1 entries   [⇐] [←] [1] [→] [⇒]

Peering Rules

[★ Create Peering Rule]

Search: [   ]

# ▲	Callee Prefix	Callee Pattern	Callee Pattern	Description	
2				Default Rule	

Showing 1 to 1 of 1 entries   [⇐] [←] [1] [→] [⇒]

To configure this setting, open the *Remote Authentication* tab and edit the following three preferences:

- **peer_auth_user:** `<username for peer auth>`

- **peer_auth_pass:** `<password for peer auth>`

- **peer_auth_realm:** `<domain for peer auth>`

← Back

Access Restrictions

Number Manipulations

NAT and Media Flow Control

Remote Authentication

	Name		Value	
❷	peer_auth_user	**1**	peeruser1	
❷	peer_auth_pass	**2**	peerpass1	
❷	peer_auth_realm	**3**	testpeering.com	
❷	peer_auth_register		☐	
❷	find_subscriber_by_uuid		☐	

Session Timers

⚠ **Important**

If you do NOT authenticate against a peer host, then the caller CLI is put into the `From` and `P-Asserted-Iden tity` headers, e.g. `"+4312345" <sip:+4312345@your-domain.com>`. If you DO authenticate, then the `From` header is `"+4312345" <sip:your_peer_auth_user@your_peer_auth_realm>` (the CLI is in the Display field, the peer_auth_user in the From username and the peer_auth_realm in the From domain), and the `P-Asserted-Identity` header is as usual like `<sip:+4312345@your-domain.com>`. So for presenting the correct CLI in *CLIP no screening* scenarios, your peering provider needs to extract the correct user either from the `From Display-Name` or from the `P-Asserted-Identity URI-User`.

Tip

You will notice that these three preferences are also shown in the *Subscriber Preferences* for each subscriber. There you can override the authentication details for all peer host if needed, e.g. if every user authenticates with his own separate credentials at your peering provider.

Tip

If **peer_auth_realm** is set, the system may overwrite the Request-URI with the peer_auth_realm value of the peer when sending the call to that peer or peer_auth_realm value of the subscriber when sending a call to the subscriber. Since this is rarely a desired behavior, it is disabled by default starting with NGCP release 3.2. If you need the replacement, you should set *set_ruri_to_peer_auth_realm: 'yes'* in */etc/ngcp-config/config.yml*.

Registering at a Peering Server

Unfortunately, the credentials configured above are not yet automatically used to register the SPCE at your peer hosts. There is however an easy manual way to do so, until this is addressed.

Configure your peering servers with the corresponding credentials in */etc/ngcp-config/templates/etc/ngcp-sems/etc/reg_agent.conf.tt2*, then execute *ngcpcfg apply*.

> ⚠️ **Important**
>
> Be aware that this will force SEMS to restart, which will drop all calls.

6.6 Configuring Rewrite Rule Sets
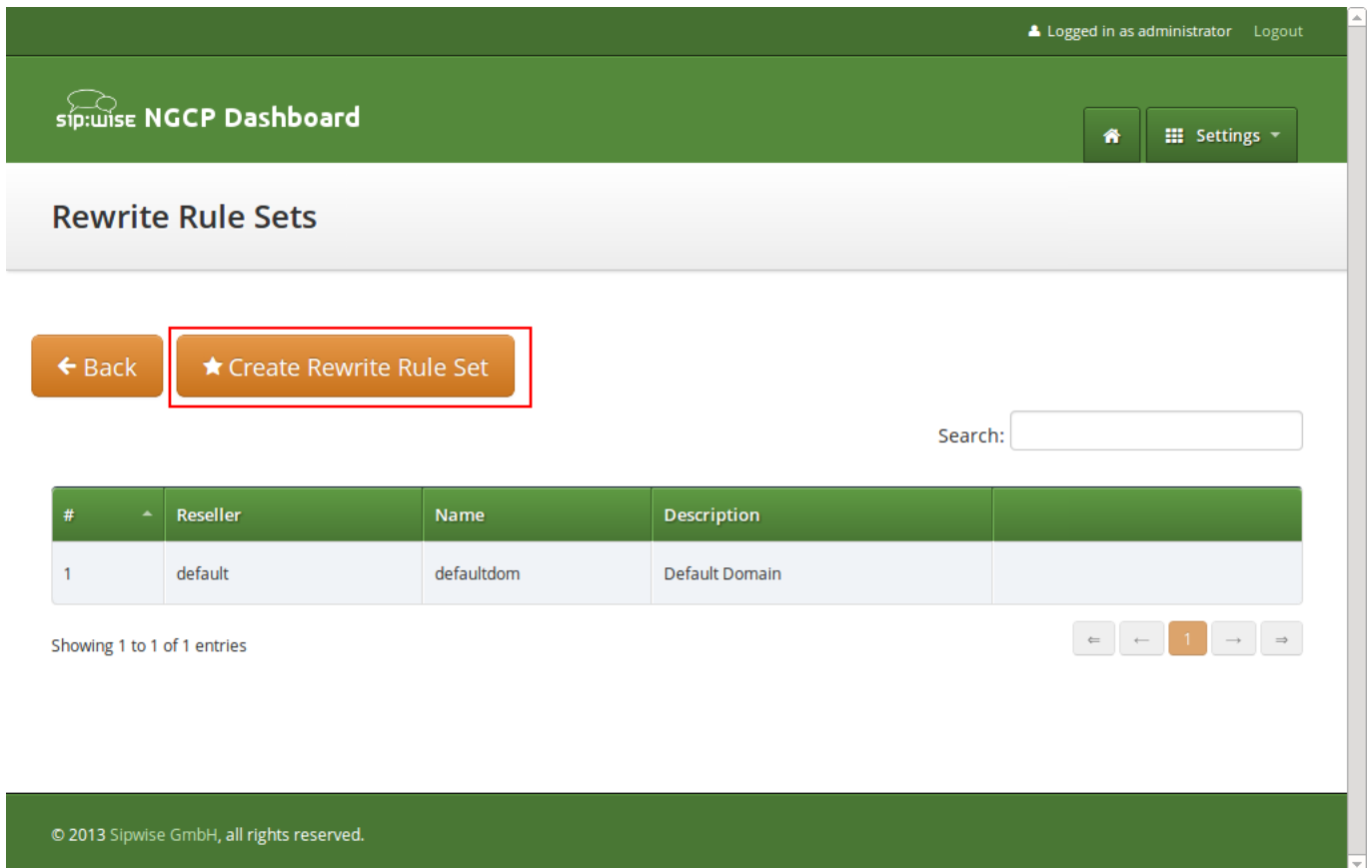
> ⚠️ **Important**
>
> On the NGCP, every phone number is treated in E.164 format *<country code><area code><subscriber number>*. Rewrite Rule Sets is a flexible tool to translate the caller and callee numbers to the proper format before the routing lookup and after the routing lookup separately. The created Rewrite Rule Sets can be assigned to the domains, subscribers and peers as a preference.

You would normally begin with creating a Rewrite Rule Set for your SIP domains. This is used to control what an end user can dial for outbound calls, and what is displayed as the calling party on inbound calls. The subscribers within a domain inherit Rewrite Rule Sets of that domain, unless this is overridden by a subscriber Rewrite Rule Set preference.

To create a new Rewrite Rule Set, go to *Settings→Rewrite Rule Sets*. There you can create a Set identified by a name. This name is later shown in your peer-, domain- and user-preferences where you can select the rule set you want to use.

Click *Create Rewrite Rule Set* and fill in the form accordingly.
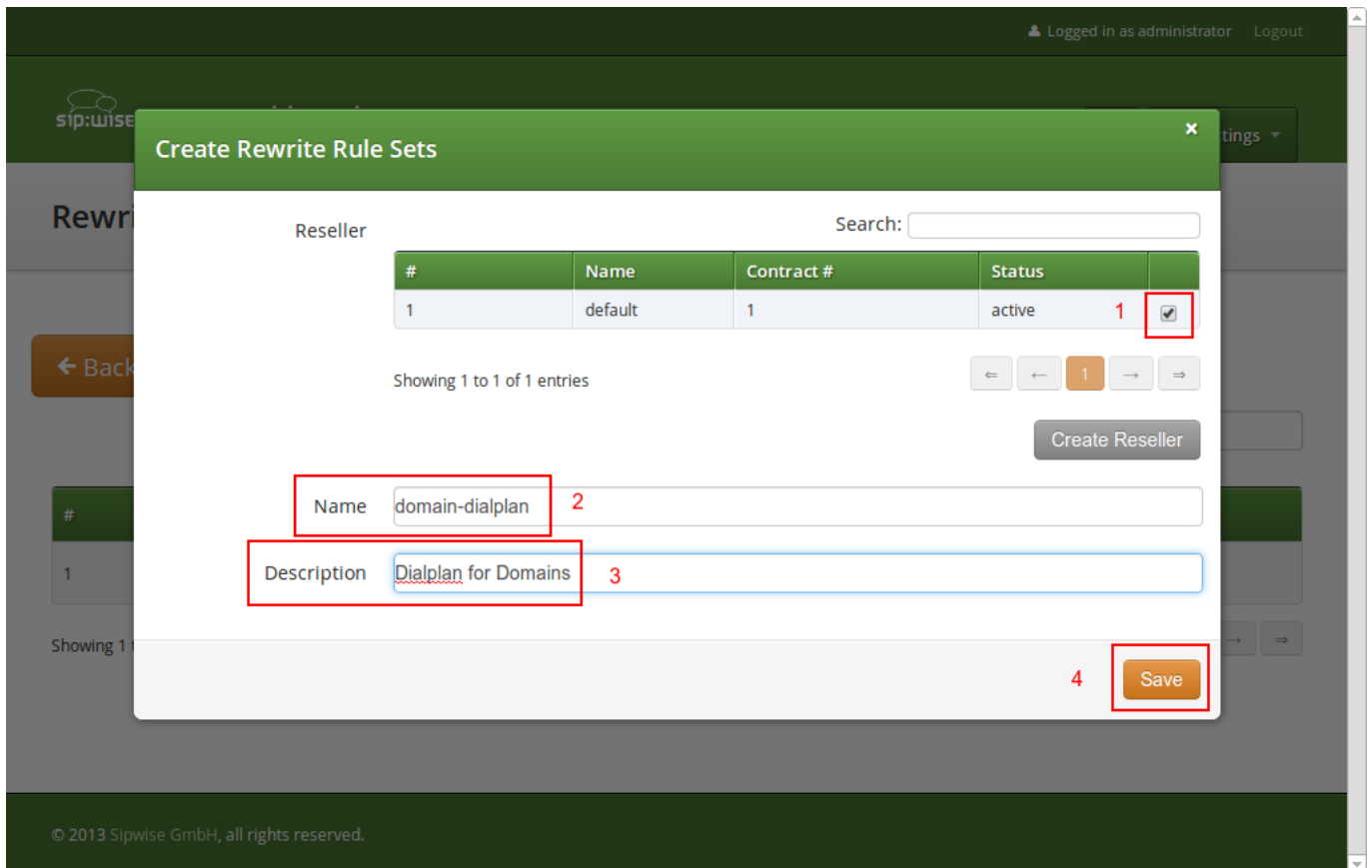
Press the *Save* button to create the set.

To view the *Rewrite Rules* within a set, hover over the row and click the *Rules* button.

The rules are ordered by *Caller* and *Callee* as well as direction *Inbound* and *Outbound*.

Tip

In Europe, the following formats are widely accepted: *+<cc><ac><sn>*, *00<cc><ac><sn>* and *0<ac><sn>*. Also, some countries allow the areacode-internal calls where only subscriber number is dialed to reach another number in the same area. Within this section, we will use these formats to show how to use rewrite rules to normalize and denormalize number formats.

6.6.1 Inbound Rewrite Rules for Caller

These rules are used to normalize user-provided numbers (e.g. passed in *From Display Name* or *P-Preferred-Identity* headers) into E.164 format. In our example, we'll normalize the three different formats mentioned above into E.164 format.

To create the following rules, click on the *Create Rewrite Rule* for each of them and fill them with the values provided below.

STRIP LEADING 00 OR +

- Match Pattern: `^(00|\+)([1-9][0-9]+)$`

- Replacement Pattern: `\2`

- Description: `International to E.164`

- Direction: `Inbound`

- Field: `Caller`

REPLACE 0 BY CALLER'S COUNTRY CODE:

- Match Pattern: `^0([1-9][0-9]+)$`

- Replacement Pattern: `${caller_cc}\1`

- Description: `National to E.164`

- Direction: `Inbound`

- Field: `Caller`

NORMALIZE LOCAL CALLS:

- Match Pattern: `^([1-9][0-9]+)$`

- Replacement Pattern: `${caller_cc}${caller_ac}\1`

- Description: `Local to E.164`

- Direction: `Inbound`

- Field: `Caller`

Normalization for national and local calls is possible with special variables `${caller_cc}` and `${caller_ac}` that can be used in Replacement Pattern and are substituted by the country and area code accordingly during the call routing.

> **Important**
>
> These variables are only being filled in when a call originates from a subscriber (because only then the cc/ac information is known by the system), so you can not use them when a calls comes from a SIP peer (the variables will be just empty in this case).

> **Tip**
>
> When routing a call, the rewrite processing is stopped after the first match of a rule, starting from top to bottom. If you have two rules (e.g. a generic one and a more specific one), where both of them would match some numbers, reorder them with the up/down arrows into the appropriate position.

Rewrite Rules for domain-dialplan

← Back   ★ Create Rewrite Rule

Rewrite rule successfully created

Inbound Rewrite Rules for Caller

	Match Pattern	Replacement Pattern	Description	
1 ↑ ↓	^(00|\+)([1-9][0-9]+)$	\2	International to E.164	
↑ ↓ 2	^0([1-9][0-9]+)$	${caller_cc}\1	National to E.164	
↑ ↓	^([1-9][0-9]+)$	${caller_cc}${caller_ac}\1	Local to E.164	

Inbound Rewrite Rules for Callee

Outbound Rewrite Rules for Caller

Outbound Rewrite Rules for Callee

6.6.2 Inbound Rewrite Rules for Callee

These rules are used to rewrite the number the end user dials to place a call to a standard format for routing lookup. In our example, we again allow the three different formats mentioned above and again normalize them to E.164, so we put in the same rules as for the caller.

STRIP LEADING 00 OR +

- Match Pattern: `^(00|\+)([1-9][0-9]+)$`

- Replacement Pattern: `\2`

- Description: `International to E.164`

- Direction: `Inbound`

- Field: `Callee`

REPLACE 0 BY CALLER'S COUNTRY CODE:

- Match Pattern: `^0([1-9][0-9]+)$`

- Replacement Pattern: `${caller_cc}\1`

- Description: `National to E.164`

- Direction: `Inbound`

- Field: `Callee`

NORMALIZE AREACODE-INTERNAL CALLS:

- Match Pattern: `^([1-9][0-9]+)$`

- Replacement Pattern: `${caller_cc}${caller_ac}\1`

- Description: `Local to E.164`

- Direction: `Inbound`

- Field: `Callee`

Tip

Our provided rules will only match if the caller dials a numeric number. If he dials an alphanumeric SIP URI, none of our rules will match and no rewriting will be done. You can however define rules for that as well. For example, you could allow your end users to dial `support` and rewrite that to your support hotline using the match pattern `^support$` and the replace pattern `43800999000` or whatever your support hotline number is.

6.6.3 Outbound Rewrite Rules for Caller

These rules are used to rewrite the calling party number for a call to an end user. For example, if you want the device of your end user to show *0<ac><sn>* if a national number calls this user, and *00<cc><ac><sn>* if an international number calls, put the following rules there.

REPLACE AUSTRIAN COUNTRY CODE 43 BY 0

- Match Pattern: `^43([1-9][0-9]+)$`

- Replacement Pattern: `0\1`

- Description: `E.164 to Austria National`

- Direction: `Outbound`

- Field: `Caller`

PREFIX 00 FOR INTERNATIONAL CALLER

- Match Pattern: `^([1-9][0-9]+)$`

- Replacement Pattern: `00\1`

- Description: `E.164 to International`

- Direction: `Outbound`

- Field: `Caller`

Tip

Note that both of the rules would match a number starting with `43`, so reorder the national rule to be above the international one (if it's not already the case).

6.6.4 Outbound Rewrite Rules for Callee

These rules are used to rewrite the called party number immediately before sending out the call on the network. This gives you an extra flexibility by controlling the way request appears on a wire, when your SBC or other device expects the called party number to have a particular tech-prefix. It can be used on calls to end users too if you want to do some processing in intermediate SIP device, e.g. apply legal intercept selectively to some subscribers.

PREFIX `sipsp#` FOR ALL CALLS

- Match Pattern: `^([0-9]+)$`

- Replacement Pattern: `sipsp#\1`

- Description: `Intercept this call`

- Direction: `Outbound`

- Field: `Callee`

6.6.5 Emergency Number Handling

Configuring Emergency Numbers is also done via Rewrite Rules.

For Emergency Calls from a subscriber to the platform, you need to define an *Inbound Rewrite Rule For Callee*, which adds a prefix `emergency_` to the number (and can rewrite the number completely as well at the same time). If the proxy detects a call to a SIP URI starting with `emergency_`, it will enter a special routing logic bypassing various checks which might make a normal call fail (e.g. due to locked or blocked numbers, insufficient credits or exceeding the max. amount of parallel calls).

TAG AN EMERGENCY CALL

- Match Pattern: `^(911|112)$`

- Replacement Pattern: `emergency_\1`

- Description: `Tag Emergency Numbers`

- Direction: `Inbound`

- Field: `Callee`

To route an Emergency Call to a Peer, you can select a specific peering group by adding a peering rule with a *callee prefix* set to `emergency_` to a peering group.

In order to normalize the emergency number to a valid format accepted by the peer, you need to assign an *Outbound Rewrite Rule For Callee*, which strips off the `emergency_` prefix. You can also use the variables `${caller_emergency_cli}`, `${caller_emergency_prefix}` and `${caller_emergency_suffix}` as well as `${caller_ac}` and `${caller_cc}`, which are all configurable per subscriber to rewrite the number into a valid format.

NORMALIZE EMERGENCY CALL FOR PEER

- Match Pattern: `^emergency_(.+)$`

- Replacement Pattern: `${caller_emergency_prefix}${caller_ac}\1`

- Description: `Normalize Emergency Numbers`

- Direction: `Outbound`

- Field: `Caller`

6.6.6 Assigning Rewrite Rule Sets to Domains and Subscribers

Once you have finished to define your Rewrite Rule Sets, you need to assign them. For sets to be used for subscribers, you can assign them to their corresponding domain, which then acts as default set for all subscribers. To do so, go to *Settings→Domains* and click *Preferences* on the domain you want the set to assign to. Click on *Edit* and select the Rewrite Rule Set created before.

You can do the same in the *Preferences* of your subscribers to override the rule on a subscriber basis. That way, you can finely control down to an individual user the dial-plan to be used. Go to *Settings→Subscribers*, click the *Details* button on the subscriber you want to edit, the click the *Preferences* button.

6.6.7 Creating Dialplans for Peering Servers

For each peering server, you can use one of the Rewrite Rule Sets that was created previously as explained in Section 6.6 (keep in mind that special variables `${caller_ac}` and `${caller_cc}` can not be used when the call comes from a peer). To do so, click on the name of the peering server, look for the preference called *Rewrite Rule Sets*.

If your peering servers don't send numbers in E.164 format *<cc><ac><sn>*, you need to create *Inbound Rewrite Rules* for each peering server to normalize the numbers for caller and callee to this format, e.g. by stripping leading + or put them from national into E.164 format.

Likewise, if your peering servers don't accept this format, you need to create *Outbound Rewrite Rules* for each of them, for example to append a + to the numbers.

7   Advanced Subscriber Configuration

The sip:provider CE provides a large amount of subscriber features in order to offer compelling VoIP services to end customers, and also to cover as many deployment scenarios as possible. In this chapter, we will go over the features and describe their behavior and their use cases.

7.1   Access Control for SIP Calls

There are two different methods to provide fine-grained call admission control to both subscribers and admins. One is *Block Lists*, where you can define which numbers or patterns can be called from a subscriber to outbound direction and which numbers or patterns are allowed to call a subscriber in inbound direction. The other is *NCOS Levels*, where the admin predefines rules for outbound calls, which are grouped in certain levels. The user can then just choose the level, or the admin can restrict a user to a certain level. Also sip:provider CE offers some options to restrict the IP addresses that subscriber is allowed to use the service from. The following chapters will discuss these features in detail.

7.1.1   Block Lists

Block Lists provide a way to control which users/numbers are able to call or to be called, based on a subscriber level, and can be found in the *Call Blockings* section of the subscriber preferences.

Block Lists are separated into *Administrative Block Lists* (*adm_block_**) and *Subscriber Block Lists* (*block_**). They both have

the same behavior, but Administrative Block Lists take higher precedence. Administrative Block Lists are only accessible by the system administrator and can thus be used to override any Subscriber Block Lists, e.g. to block certain destinations. The following break-down of the various block features apply to both types of lists.

Block Modes

Block lists can either be *whitelists* or *blacklists* and are controlled by the User Preferences *block_in_mode*, *block_out*mode__ and their administrative counterparts.

- The *blacklist* mode (option is not checked tells the system to **allow anything except the entries in the list**. This mode is used if you want to just block certain numbers and allow all the rest.

- The *whitelist* mode indicates to **reject anything except the entries in the list**. This is used if you want to enforce a strict policy and allow only selected destinations or sources.

You can change a list mode from one to the other at any time.

Block Lists

The list contents are controlled by the User Preferences *block_in_list*, *block_out_list* and their administrative counterparts. Click on the *Edit* button in the *Preferences* view to define the list entries.

In block list entries, you can provide shell patterns like ` * ` and `[]`. The behavior of the list is controlled by the *block_xxx_mode* feature (so they are either allowed or rejected). In our example above we have *block_out_mode* set to *blacklist*, so all calls to US numbers and to the Austrian number +431234567 are going to be rejected.
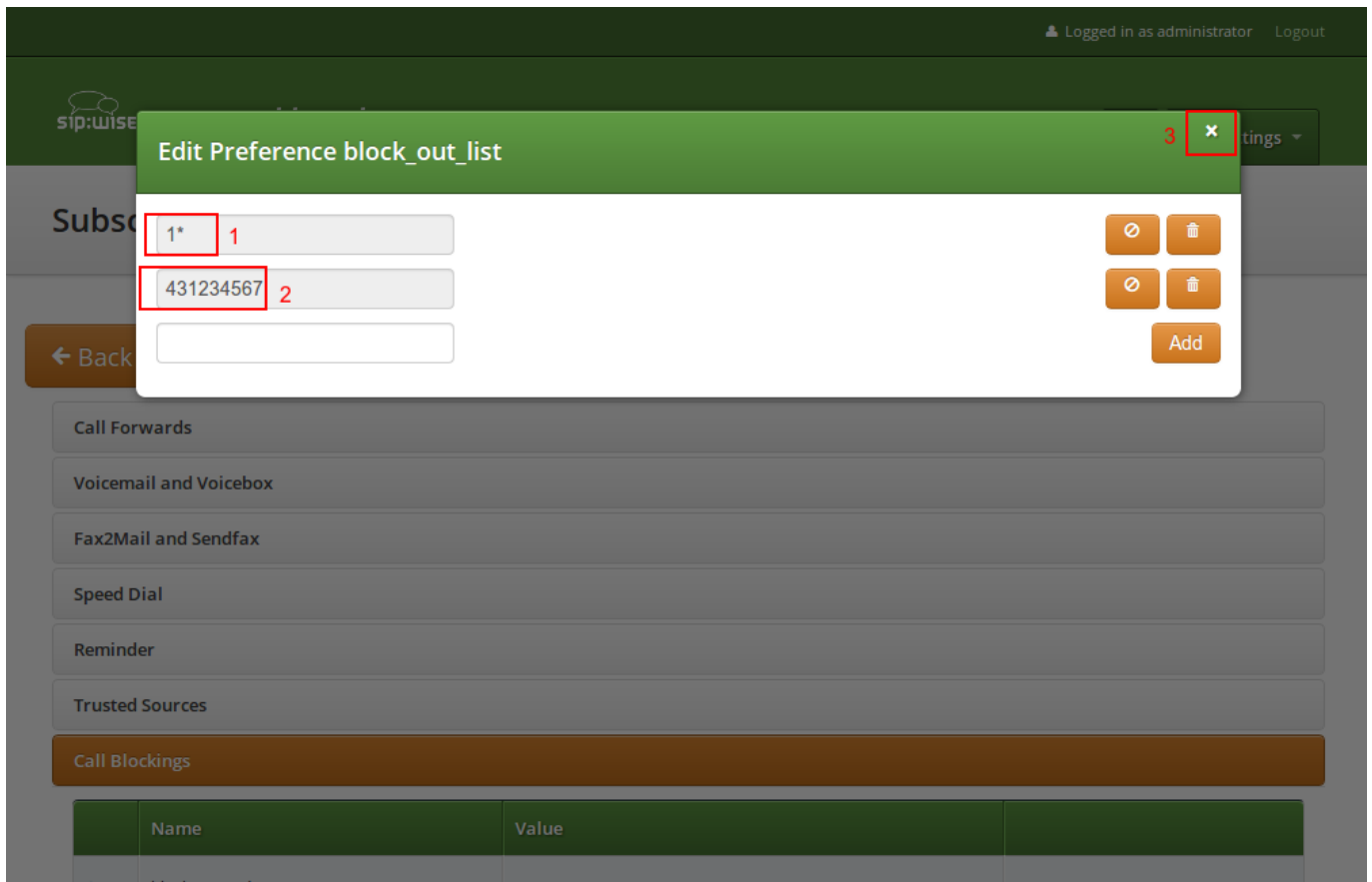
Click the *Close* icon once you're done editing your list.

Block Anonymous Numbers

For incoming call, the User Preference *block_in_clir* and *adm_block_in_clir* controls whether or not to reject incoming calls with number supression (either "[Aa]nonymous" in the display- or user-part of the From-URI or a header *Privacy: id* is set). This flag is independent from the Block Mode.

7.1.2 NCOS Levels
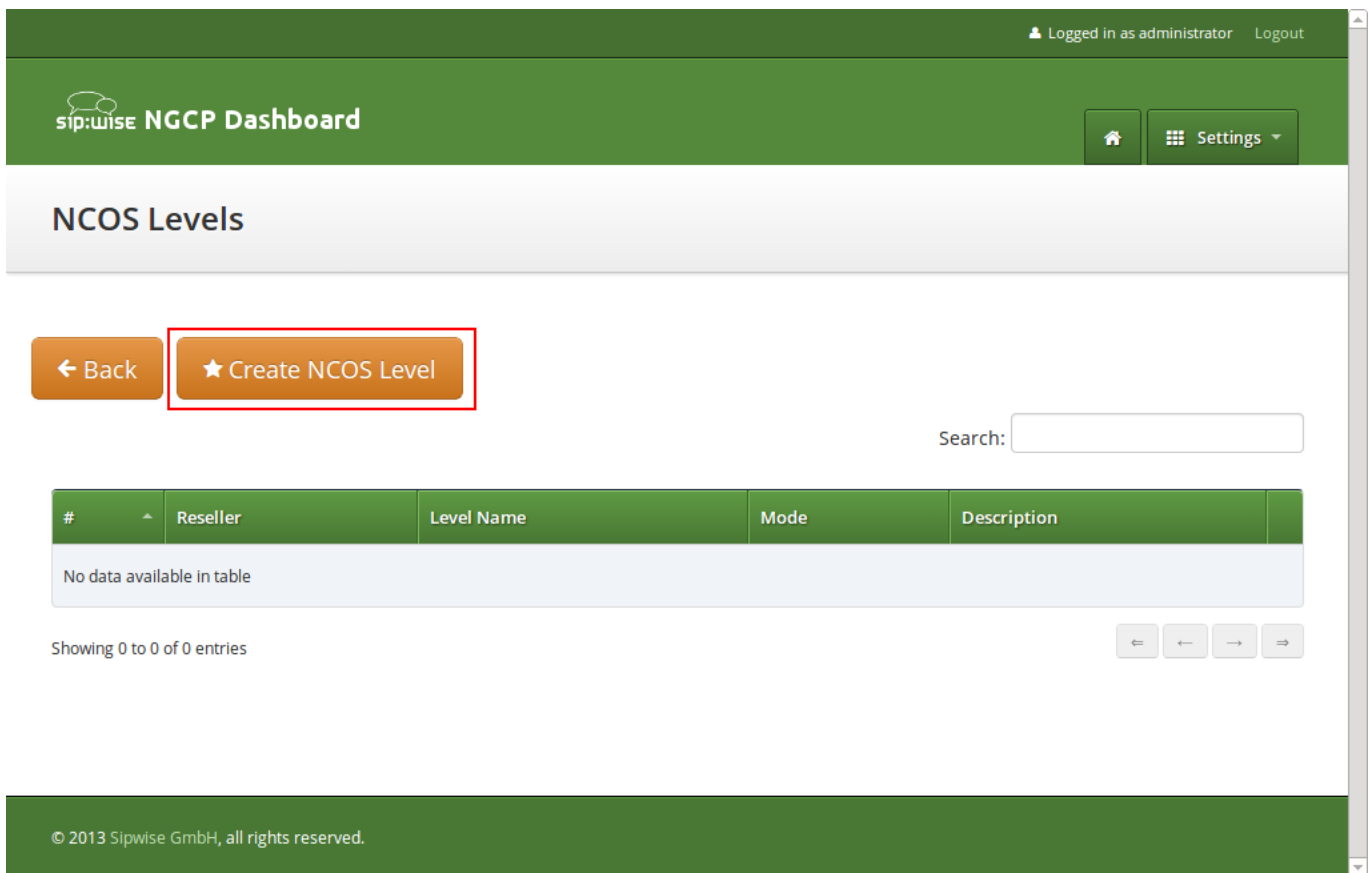
NCOS Levels provide predefined lists of allowed or denied destinations for outbound calls of local subscribers. Compared to *Block Lists*, they are much easier to manage, because they are defined on a global scope, and the individual levels can then be assigned to each subscriber. Again there is the distinction for user- and administrative-levels.

NCOS levels can either be *whitelists* or *blacklists*.

- The *blacklist* mode indicates to **allow everything except the entries in this level**. This mode is used if you want to just block certain destinations and allow all the rest.

- The *whitelist* mode indicates to **reject anything except the entries in this level**. This is used if you want to enforce a strict policy and allow only selected destinations.

Creating NCOS Levels

To create an NCOS Level, go to *Settings→NCOS Levels* and press the *Create NCOS Level* button.

Select a reseller, enter a name, select the mode and add a description, then click the *Save* button.

Creating Rules per NCOS Level

To define the rules within the newly created NCOS Level, click on the *Patterns* button of the level.

In the *Number Patterns* view you can create multiple patterns to define your level, one after the other. Click on the *Create Pattern Entry* Button on top and fill out the form.

In this example, we block (since the mode of the level is *blacklist*) all numbers starting with 439. Click the *Save* button to save the entry in the level.

The option *include local area code in list* for a blacklist means that calls within the area code of the subscribers are denied, and for whitelist that they are allowed, respectively. For example if a subscriber has country-code 43 and area-code 1, then selecting this checkbox would result in an implicit entry ˆ431.

Assigning NCOS Levels to Subscribers/Domains

Once you've defined your NCOS Levels, you can assign them to local subscribers. To do so, navigate to *Settings→Subscribers*, search for the subscriber you want to edit, press the *Details* button and go to the *Preferences* View. There, press the *Edit* button on either the *ncos* or *adm*ncos *setting in the* Call Blockings__ section.

You can assign the NCOS level to all subscribers within a particular domain. To do so, navigate to *Settings→Domains*, select the domain you want to edit and click *Preferences*. There, press the *Edit* button on either *ncos* or *admin_ncos* in the *Call Blockings* section.

Note: if both domain and subscriber have same NCOS preference set (either *ncos* or *adm_ncos*, or both) the subscriber's preference is used. This is done so that you can override the domain-global setting on the subscriber level.

7.1.3   IP Address Restriction

The sip:provider CE provides subscriber preference *allowed_ips* to restrict the IP addresses that subscriber is allowed to use the service from. If the REGISTER or INVITE request comes from an IP address that is not in the allowed list, the sip:provider CE will reject it with a 403 message. Also a voice message can be played when the call attempt is rejected (if configured).

By default, *allowed_ips* is an empty list which means that subscriber is not restricted. If you want to configure a restriction, navigate to *Settings→Subscribers*, search for the subscriber you want to edit, press *Details* and then *Preferences* and press *Edit* for the *allowed_ips* preference in the *Access Restrictions* section.

Press the Edit button to the right of empty drop-down list.

You can enter multiple allowed IP addresses or IP address ranges one after another. Click the *Add* button to save each entry in the list. Click the *Delete* button if you want to remove some entry.

7.2   Call Forwarding and Call Hunting

The sip:provider CE provides the capabilities for normal *call forwarding* (deflecting a call for a local subscriber to another party immediately or based on events like the called party being busy or doesn't answer the phone for a certain number of seconds) and *serial call hunting* (sequentially executing a group of deflection targets until one of them succeeds). Targets can be stacked, which means if a target is also a local subscriber, it can have another call forward or hunt group which is executed accordingly.

Call Forwards and Call Hunting Groups can either be executed unconditionally or based on a *Time Set Definition*, so you can define deflections based on time period definitions (e.g. Monday to Friday 8am to 4pm etc).

7.2.1   Setting a simple Call Forward

Go to your *Subscriber Preferences* and click *Edit* on the Call Forward Type you want to set (e.g. *Call Forward Unconditional*.

If you select *URI/Number* in the *Destination* field, you also have to set a *URI/Number*. The timeout defines for how long this destination should be tried to ring.

7.2.2 Advanced Call Hunting

If you want multiple destinations to be executed one after the other, you need to change into the *Advanced View* when editing your call forward. There, you can select multiple *Destination Set*/*Time Set* pairs to be executed.

A *Destination Set* is a list of destinations to be executed one after another.

A *Time Set* is a time definition when to execute this *Destination Set*.

Configuring Destination Sets

Click on *Manage Destination Sets* to see a list of available sets. The *quickset_cfu* has been implicitly created during our creation of a simple call forward. You can edit it to add more destinations, or you can create a new destination set.

When you close the *Destination Set* Overview, you can now assign your new set in addition or instead of the *quickset_cfu* set.

Press *Save* to store your settings.

Configuring Time Sets

Click on *Manage Time Sets* in the advanced call-forward menu to see a list of available time sets. By default there are none, so you have to create one.

You need to provide a *Name*, and a list of *Periods* where this set is active. If you only set the top setting of a date field (like the *Year* setting in our example above), then it's valid for just this setting (like the full year of *2013* in our case). If you provide the bottom setting as well, it defines a period (like our *Month* setting, which means from beginning of April to end of September).

> **Important**
>
> the period is a *through* definition, so it covers the full range. If you define an *Hour* definition *8-16*, then this means from *08:00* to *16:59:59* (unless you filter the *Minutes* down to something else).

If you close the *Time Sets* management, you can assign your new time set to the call forwards you're configuring.

7.3 Limiting Subscriber Preferences via Subscriber Profiles

The preferences a subscriber can provision by himself via the CSC can be limited via profiles within profile sets assigned to subscribers.

7.3.1 Subscriber Profile Sets

Profile sets define containers for profiles. The idea is to define profile sets with different profiles by the administrator (or the reseller, if he is permitted to do so). Then, a subscriber with administrative privileges can re-assign profiles within his profile sets for the subscribers of his customer account.

Profile Sets can be defined in *Settings→Subscriber Profiles*. To create a new Profile Set, click *Create Subscriber Profile Set*.

You need to provide a reseller, name and description.

To create Profiles within a Profile Set, hover over the Profile Set and click the *Profiles* button.

Profiles within a Profile Set can be created by clicking the *Create Subscriber Profile* button.

Checking the *Default Profile* option causes this profile to get assigned automatically to all subscribers, who have the profile set assigned. Other options define the user preferences which should be made available to the subscriber.

7.4 Voicemail System

7.4.1 Accessing the IVR Menu

For a subscriber to manage his voicebox via IVR, there are two ways to access the voicebox. One is to call the URI `voicebox@yourdomain` from the subscriber itself, allowing password-less access to the IVR, as the authentication is already done on SIP level. The second is to call the URI `voiceboxpass@yourdomain` from any subscriber, causing the system to prompt for a mailbox and a PIN.

Mapping numbers and codes to IVR access

Since access might need to be provided from external networks like PSTN/Mobile, and since certain SIP phones don't support calling alphanumeric numbers to dial `voicebox`, you can map any arbitrary number to the voicebox URIs using rewrite rules.

To do so, you can provision a match pattern like `^(00|\+)12345$` with a replace pattern `voicebox` or `voiceboxpass` to

map a number to either password-less or password-based IVR access.

External IVR access

When reaching `voiceboxpass`, the subscriber is prompted for her mailbox number and a password. All numbers assigned to a subscriber are valid input (primary number and any alias number). By default, the required format is in E.164, so the subscriber needs to enter the full number including country code, for example `4912345` if she got assigned a German number.

You can globally configure a rewrite rule in `config.yml` using `asterisk.voicemail.normalize_match` and `asterisk.voicemail.normalize_replace`, allowing you to customize the format a subscriber can enter, e.g. having `^0([1-9][0-9]+)$` as match part and `49$1` as replace part to accept German national format.

7.4.2 IVR Menu Structure

The following list shows you how the voicebox menu is structured.

- *1* Read voicemail messages

  - *3* Advanced options

    * *3* To Hear messages Envelope
    * *** Return to the main menu

  - *4* Play previous message

  - *5* Repeat current message

  - *6* Play next message

  - *7* Delete current message

  - *9* Save message in a folder

    * *0* Save in new Messages
    * *1* Save in old Messages
    * *2* Save in Work Messages
    * *3* Save in Family Messages
    * *4* Save in Friends Messages
    * *#* Return to the main menu

- *2* Change folders

  - *0* Switch to new Messages

  - *1* Switch to old Messages

  - *2* Switch to Work Messages

  - *3* Switch to Family Messages

  - *4* Switch to Friends Messages

  - *#* Get Back

- *3* Advanced Options

  - *** To return to the main menu

- *0* Mailbox options

  - *1* Record your unavailable message

    - *** *1* accept it

    - *** *2* Listen to it

    - *** *3* Rerecord it

  - *2* Record your busy message

    - *** *1* accept it

    - *** *2* Listen to it

    - *** *3* Rerecord it

  - *3* Record your name

    - *** *1* accept it

    - *** *2* Listen to it

    - *** *3* Rerecord it

  - *4* Record your temporary greetings

    - *** *1* accept it

    - *** *2* Listen to it

    - *** *3* Rerecord it

  - *5* Change your password

  - *** To return to the main menu

- *** Help

- *#* Exit

7.5   Configuring Sound Sets

The sip:provider CE provides the administrator with ability to upload the voice prompts for typical call error announcements on the *Sound Sets page*. There is a preference *sound_set* on Domain and Subscriber levels to like subscribers to the sound set that they should hear (as usual the subscriber preference overrides the domain one). Sound Sets can be defined in *Settings→Sound Sets*. To create a new Sound Set, click *Create Sound Set*. Then click the *Files* button.

7.5.1   Early Reject Sound Sets

The failed call announcements are grouped under *Early Rejects* section. Unfold the section and click *Upload* next to the sound handles (Names) that you want to use. Choose a WAV file from your file system, and click the Loopplay setting if you want to play the file in a loop instead of just once. Click Save to upload the file. If you don't provide the sound files for any handles they will not be used and sip:provider CE will adhere to the default behavior of sending the error response code back to the user.

Note

You can use 8 or 16 bit mono WAV audio files.

Table 2: Early Reject Sound Sets

Handle	Description
block_in	This is what the calling party hears when a call is made from a number that is blocked by the incoming block list (*adm_block_in_list*, *block_in_list* subscriber preferences)
block_out	This is what the calling party hears when a call is made to a number that is blocked by the outgoing block list (*adm_block_out_list*, *block_out_list* subscriber preferences)
block_ncos	This is what the calling party hears when a call is made to a number that is blocked by the NCOS level assigned to the subscriber or domain (the NCOS level chosen in *ncos* and *adm_ncos* preferences)
block_override_pin_wrong	Announcement played to calling party if it used wrong PIN code to override the outgoing user block list or the NCOS level for this call (the PIN set by *block_out_override_pin* and *adm_block_out_override_pin* preferences)
locked_in	Announcement played on incoming call to a subscriber that is locked for incoming calls
locked_out	Announcement played on outgoing call to subscriber that is locked for outgoing calls
max_calls_in	Announcement played on incoming call to a subscriber who has exceeded the *concurrent_max* limit by sum of incoming and outgoing calls or whose customer has exceeded the *concurrent_max_per_account* limit by sum of incoming and outgoing calls
max_calls_out	Announcement played on outgoing call to a subscriber who has exceeded the *concurrent_max* (total limit) or *concurrent_max_out* (limit on number of outbound calls) or whose customer has exceeded the *concurrent_max_per_account* or *concurrent_max_out_per_account* limit
max_calls_peer	Announcement played on calls from the peering if that peer has reached the maximum number of concurrent calls (configured by admin in *concurrent_max* preference of peering server)
unauth_caller_ip	This is what the calling party hears when it tries to make a call from unauthorized IP address or network (*allowed_ips*, *man_allowed_ips* preferences)

Table 2: (continued)

Handle	Description
relaying_denied	Announcement played on inbound call from trusted IP (e.g. external PBX) with non-local Request-URI domain
invalid_speeddial	This is what the calling party hears when it calls an empty speed-dial slot
cf_loop	Announcement played when the called subscriber has the call forwarding configured to itself
callee_offline	Announcement played on incoming call to the subscriber which is currently not registered
callee_busy	Announcement played on incoming call to the subscriber which is currently busy (486 response from the UAS)
callee_unknown	Announcement that is played on call to unknown or invalid number (not associated with any of our subscribers/hunt groups)
callee_tmp_unavailable	Announcement played on incoming call to the subscriber which is currently unavailable (408, other 4xx or no response code or 30x with malformed contact)
peering_unavailable	Announcement played in case of outgoing off-net call when there is no peering rule matching this destination and/or source
voicebox_unavailable	Announcement played on call to voicebox if the voicemail server is not configured (system operation is impaired)
emergency_unsupported	Announcement played when emergency destination is dialed but the emergency calls are administratively prohibited for this user or domain (*reject_emergency* preference is enabled)
no_credit	Announcement played when prepaid account has insufficient balance to make a call to this destination

8   Customer Self-Care Interfaces

There are two ways for end users to maintain their subscriber settings: via the *Customer Self-Care Web Interface* and via *Vertical Service Codes* using their SIP phones.

8.1   The Customer Self-Care Web Interface

The NGCP provides a web panel for end users (CSC panel) to maintain their subscriber accounts, which is running on *https://<ce-ip>*. Every subscriber can log in there, change subscriber feature settings, view their call lists, retrieve voicemail messages and trigger calls using the click-to-dial feature.

8.1.1   Login Procedure

To log into the CSC panel, the end user has to provide his full web username (e.g. `user1@1.2.3.4`) and the web password defined in Section 6.2. Once logged in, he can change his web password in the *Account* section. This will NOT change his SIP password, so if you control the end user devices, you can auto-provision the SIP password into the device and keep it secret, and just hand over the web password to the customer. This way, the end user will only be able to place calls with this auto-provisioned device and not with an arbitrary soft-phone, but can nonetheless manage his account via the CSC panel.

> **Important**
>
> You can simplify the login procedure for one SIP domain in such a way that users in this domain only need to pass the user part (e.g. `user1`) as a username instead of the full web username to log in by setting the parameter *www_csc→site_domain* in the config file */etc/ngcp-config/config.yml* to the corresponding domain (e.g. `1.2.3.4`) and execute `ngcpcfg apply`.

8.1.2   Site Customization

As an operator, you can change the appearance of the CSC panel by modifying a couple of parameters in the section *www_csc→site_config* of the config file */etc/ngcp-config/config.yml*. Modify the site title, your company details and the logo to reflect your use case.

You can also enable/disable specific languages a user can choose from in the CSC panel. Currently, English (`en`), French (`fr`), German (`de`) and Spanish (`es`) are supported and English is activated by default.

After changing one or more of the parameters in this file, execute `ngcpcfg apply` to activate the changes.

8.2   The Vertical Service Code Interface

Vertical Service Codes (VSC) are codes a user can dial on his phone to provision specific features for his subscriber account. The format is `*<code>*<value>` to activate a specific feature, and `#<code>` or `#<code>#` to deactivate it. The *code* parameter is a two-digit code, e.g. `72`. The *value* parameter is the value being set for the corresponding feature.

> **!** **Important**
>
> The *value* user input is normalized using the Rewrite Rules Sets assigned to domain as described in Section 6.6.

By default, the following codes are configured for setting features. The examples below assume that there is a domain rewrite rule normalizing the number format *0<ac><sn>* to *<cc><ac><sn>* using *43* as country code.

- **72** - enable *Call Forward Unconditional* e.g. to 431000 by dialing `*72*01000`, and disable it by dialing `#72`.

- **90** - enable *Call Forward on Busy* e.g. to 431000 by dialing `*90*01000`, and disable it by dialing `#90`.

- **92** - enable *Call Forward on Timeout* e.g. after 30 seconds of ringing to 431000 by dialing `*92*30*01000`, and disable it by dialing `#92`.

- **93** - enable *Call Forward on Not Available* e.g. to 431000 by dialing `*93*01000`, and disable it by dialing `#93`.

- **50** - set *Speed Dial Slot*, e.g. set slot 1 to 431000 by dialing `*50*101000`, which then can be used by dialing `*1`.

- **55** - set *One-Shot Reminder Call* e.g. to 08:30 by dialing `*55*0830`.

- **31** - set *Calling Line Identification Restriction* for one call, e.g. to call 431000 anonymously dial `*31*01000`.

- **80** - call using *Call Block Override PIN*, number should be prefixed with a block override PIN configured in admin panel to disable the outgoing user/admin block list and NCOS level for a call. For example, when override PIN is set to 7890, dial `*80*789001000` to call 431000 bypassing block lists.

You can change any of the codes (but not the format) in */etc/ngcp-config/config.yml* in the section *sems→vsc*. After the changes, execute `ngcpcfg apply`.

> **!** **Caution**
>
> If you have the EMTAs under your control, make sure that the specified VSCs don't overlap with EMTA-internal VSCs, because the VSC calls must be sent to the NGCP via SIP like normal telephone calls.

8.3 The Voicemail Interface

NGCP offers several ways to access the Voicemail box.

The CSC panel allows your users to listen to voicemail messages from the web browser, delete them and call back the user who left the voice message. User can setup voicemail forwarding to the external email and the PIN code needed to access the voicebox from any telephone also from the CSC panel.

To manage the voice messages from SIP phone: simply dial internal voicemail access number `2000`.

To change the access number: look for the parameter *voicemail_number* in */etc/ngcp-config/config.yml* in the section *sems→vsc*. After the changes, execute `ngcpcfg apply`.

Tip

To let the callers leave a voice message when user is not available he should enable Call Forward to Voicebox. The Call Forward can be provisioned from the CSC panel as well as by dialing Call Forward VSC with the voicemail number. E.g. when parameter *voicemail_number* is set to *9999*, a Call Forward on Not Available to the Voicebox is set if the user dials *93*9999. As a result, all calls will be redirected to the Voicebox if SIP phone is not registered.

To manage the voice messages from any phone:

- As an operator, you can setup some DID number as external voicemail access number: for that, you should add a special rewrite rule (Inbound Rewrite Rule for Callee, see Section 6.6.) on the incoming peer, to rewrite that DID to "voiceboxpass". Now when user calls this number the call will be forwarded to the voicemail server and he will be prompted for mailbox and password. The mailbox is the full E.164 number of the subscriber account and the password is the PIN set in the CSC panel.

- The user can also dial his own number from PSTN, if he setup Call Forward on Not Available to the Voicebox, and when reaching the voicemail server he can interrupt the "user is unavailable" message by pressing * key and then be prompted for the PIN. After entering PIN and confirming with # key he will enter own voicemail menu. PIN is random by default and must be kept secret for that reason.

9 Billing Configuration

This chapter describes the steps necessary to rate calls and export rated CDRs (call detail records) to external systems.

9.1 Billing Data Import

Service billing on the NGCP is based on billing profiles, which may be assigned to VoIP accounts and SIP peerings. The design focuses on a simple, yet flexible approach, to support arbitrary dial-plans without introducing administrative overhead for the system administrators. The billing profiles may define a base fee and free time or free money per billing interval. Unused free time or money automatically expires at the end of the billing interval.

Each profile may have call destinations (usually based on E.164 number prefix matching) with configurable fees attached. Call destination fees each support individual intervals and rates, with a different duration and/or rate for the first interval. (e.g.: charge the first minute when the call is opened, then every 30 seconds, or make it independent of the duration at all) It is also possible to specify different durations and/or rates for peak and off-peak hours. Peak time may be specified based on weekdays, with additional support for manually managed dates based on calendar days. The call destinations can finally be grouped for an overview on user's invoices by specifying a zone in two detail levels. (E.g.: national landline, national mobile, foreign 1, foreign 2, etc.)

9.1.1 Creating Billing Profiles

The first step when setting up billing data is to create a billing profile, which will be the container for all other billing related data. Go to *Settings→Billing* and click on *Create Billing Profile*.

The fields *Reseller*, *Handle* and *Name* are mandatory.

- **Reseller**: The reseller this billing profile belongs to.

- **Handle**: A unique, permanently fixed string which is used to attach the billing profile to a VoIP account or SIP peering contract.

- **Name**: A free form string used to identify the billing profile in the *Admin Panel*. This may be changed at any time.

- **Interval charge**: A base fee for the billing interval, specifying a monetary amount (represented as a floating point number) in whatever currency you want to use.

- **Interval free time**: If you want to include free calling time in your billing profile, you may specify the number of seconds that are available every billing interval. See *Creating Billing Fees* below on how to select destinations which may be called using the free time.

- **Interval free cash**: Same as for *interval free time* above, but specifies a monetary amount which may be spent on outgoing calls. This may be used for example to implement a minimum turnover for a contract, by setting the *interval charge* and *interval free cash* to the same values.

- **Fraud monthly limit**: The monthly fraud detection limit (in Cent) for accounts with this billing profile. If the call fees of an account reach this limit within a billing interval, an action can be triggered.

- **Fraud monthly lock**: a choice of *none, foreign, outgoing, incoming, global*. Specifies a lock level which will be used to lock the account and his subscribers when *fraud monthly limit* is exceeded.

- **Fraud monthly notify**: An email address or comma-separated list of email addresses that will receive notifications when *fraud monthly limit* is exceeded.

- **Fraud daily limit**: The fraud detection limit (in Cent) for accounts with this billing profile. If the call fees of an account reach this limit within a calendar day, an action can be triggered.

- **Fraud daily lock**: a choice of *none, foreign, outgoing, incoming, global*. Specifies a lock level which will be used to lock the account and his subscribers when *fraud daily limit* is exceeded.

- **Fraud daily notify**: An email address or comma-separated list of email addresses that will receive notifications when *fraud daily limit* is exceeded.

- **Currency**: The currency symbol for your currency. Any UTF-8 character may be used and will be printed in web interfaces.

- **VAT rate**: The percentage of value added tax for all fees in the billing profile. Currently for informational purpose only and not used further.

- **VAT included**: Whether VAT is included in the fees entered in web forms or uploaded to the platform. Currently for informational purpose only and not used further.

9.1.2 Creating Billing Fees

Each *Billing Profile* holds multiple *Billing Fees*.

To set up billing fees, click on the *Fees* button of the billing profile you want to configure. Billing fees may be uploaded using a configurable CSV file format, or entered directly via the web interface by clicking *Create Fee Entry*. To configure the CSV field order for the file upload, rearrange the entries in the *www_admin→fees_csv→element_order* array in */etc/ngcp-config/config.yml* and execute the command `ngcpcfg apply`. For input via the web interface, just fill in the text fields accordingly.

In both cases, the following information may be specified independently for every destination:

- **Zone**: A zone for a group of destinations. May be used to group destinations for simplified display, e.g. on invoices. (e.g. `foreign zone 1`)

- **Source**: The source pattern. This is a POSIX regular expression matching the complete source URI (e.g. `^.*@sip\.example\.org$` or `^someone@sip\.sipwise\.com$` or just `.` to match everything). If you leave this field empty, the default pattern `.` matching everything will be set implicitly. Internally, this pattern will be matched against the `<source_cli>@<source_domain>` fields of the CDR.

- **Destination**: The destination pattern. This is a POSIX regular expression matching the complete destination URI (e.g. `someone@sip\.example\.org` or `^43`). This field must be set.

- **Direction**: `Outbound` for standard origination fees (applies to callers placing a call and getting billed for that) or `Inbound` for termination fees (applies to callees if you want to charge them for receiving various calls, e.g. for 800-numbers). *If in doubt, use Outbound*. If you upload fees via CSV files, use `out` or `in`, respectively.

> ⚠ **Important**
>
> The {source, destination, direction} combination needs to be unique for a billing profile. The system will return an error if such a set is specified twice, both for the file upload and the input via the web interface.

> ⚠ **Important**
>
> There are several internal services (vsc, conference, voicebox) which will need a specific destination entry with a domain-based destination. If you don't want to charge the same (or nothing) for those services, add a fee for destination `\.local$` there. If you want to charge different amounts for those services, break it down into separate fee entries for `@vsc\.local$`, `@conference\.local$` and `@voicebox\.local$` with the according fees. **NOT CREATING EITHER THE CATCH-ALL FEE OR THE SEPARATE FEES FOR THE `.local` DOMAIN WILL BREAK YOUR RATING PROCESS!**

- **Onpeak init rate**: The rate for the first rating interval in cent (of whatever currency, represented as a floating point number) per second. Applicable to calls during onpeak hours.

- **Onpeak init interval**: The duration of the first billing interval, in seconds. Applicable to calls during onpeak hours.

- **Onpeak follow rate**: The rate for subsequent rating intervals in cent (of whatever currency, represented as a floating point number) per second. Applicable to calls during onpeak hours. Defaults to *onpeak init rate*.

- **Onpeak follow interval**: The duration of subsequent billing intervals, in seconds. Applicable to calls during onpeak hours. Defaults to *onpeak init interval*.

- **Offpeak init rate**: The rate for the first rating interval in cent (of whatever currency, represented as a floating point number) per second. Applicable to calls during off-peak hours. Defaults to *onpeak init rate*.

- **Offpeak init interval**: The duration of the first billing interval, in seconds. Applicable to calls during off-peak hours. Defaults to *onpeak init interval*.

- **Offpeak follow rate**: The rate for subsequent rating intervals in cent (of whatever currency, represented as a floating point number) per second. Applicable to calls during off-peak hours. Defaults to *offpeak init rate* if that one is specified, or to *onpeak follow rate* otherwise.

- **Offpeak follow interval**: The duration of subsequent billing intervals, in seconds. Applicable to calls during off-peak hours. Defaults to *offpeak init interval* if that one is specified, or to *onpeak follow interval* otherwise.

- **Use free time**: Specifies whether free time minutes may be used when calling this destination. May be specified in the file upload as 0, n[o], f[alse] and 1, y[es], t[rue] respectively.

9.1.3   Creating Off-Peak Times

To be able to differentiate between on-peak and off-peak calls, the platform stores off-peak times for every billing profile based on weekdays and/or calendar days. To edit the settings for a billing profile, go to *Settings→Billing* and press the *Peaktimes* button on the billing profile you want to configure.

To set off-peak times for a weekday, click on *Edit* next to the according weekday. You will be presented with two input fields which both receive a timestamp in the form of *hh:mm:ss* specifying a time of day for the start and end of the off-peak period. If any of the fields is left empty, the system will automatically insert 00:00:00 (*start* field) or *23:59:59* (*end* field). Click on *Add* to store the setting in the database. You may create more than one off-peak period per weekday. To delete a range, just click *Delete* next to the entry. Click the *close* icon when done.

To specify off-peak ranges based on calendar dates, click on *Create Special Off-Peak Date*. Enter a date in the form of *YYYY-*

MM-DD hh:mm:ss into the *Start Date/Time* input field and *End Date/Time* input field to define a range for the off-peak period.

9.1.4 Fraud Detection and Locking

The NGCP supports a fraud detection feature, which is designed to detect accounts causing unusually high customer costs, and then to perform one of several actions upon those accounts. This feature can be enabled and configured through two sets of billing profile options described in Section 9.1.1, namely the monthly (*fraud monthly limit*, *fraud monthly lock* and *fraud monthly notify*) and daily limits (*fraud daily limit*, *fraud daily lock* and *fraud daily notify*). Either monthly/daily limits or both of them can be active at the same time.

Once a day, shortly after midnight local time, a background script automatically checks all accounts which are linked to a billing profile enabled for fraud detection, and selects those which have caused a higher cost than the *fraud monthly limit* configured in the billing profile, within the currently active billing interval (e.g. in the current month), or a higher cost than the *fraud daily limit* configured in the billing profile, within the calendar day. It then proceeds to perform at least one of the following actions on those accounts:

- If **fraud lock** is set to anything other than *none*, it will lock the account accordingly (e.g. if **fraud lock** is set to *outgoing*, the account will be locked for all outgoing calls).

- If anything is listed in **fraud notify**, an email will be sent to the email addresses configured. The email will contain information about which account is affected, which subscribers within that account are affected, the current account balance and the configured fraud limit, and also whether or not the account was locked in accordance with the **fraud lock** setting. It should be noted that this email is meant for the administrators or accountants etc., and not for the customer.

> **!** **Important**
>
> You can override these settings on a per-account basis via SOAP or the Admin interface.

> **!** **Caution**
>
> Accounts that were automatically locked by the fraud detection feature will **not** be automatically unlocked when the next billing interval starts. This has to be done manually through the administration panel or through the provisioning interface.

> **!** **Important**
>
> If fraud detection is configured to only send an email and not lock the affected accounts, it will continue to do so for over-limit accounts every day. The accounts must either be locked in order to stop the emails (only currently active accounts are considered when the script looks for over-limit accounts) or some other action to resolve the conflict must be taken, such as disabling fraud detection for those accounts.

9.2   Billing Data Export

Regular billing data export is done using CSV (*comma separated values*) files which may be downloaded from the platform using the *cdrexport* user which has been created during the installation.

There are two types of exports. One is *CDR* (Call Detail Records) used to charge for calls made by subscribers, and the other is *EDR* (Event Detail Records) used to charge for provisioning events like enabling certain features.

9.2.1   File Name Format

In order to be able to easily identify billing files, the file names are constructed by the following fixed-length fields:

```
<prefix><separator><version><separator><timestamp><separator><sequence number>< ↩
    suffix>
```

The definition of the specific fields is as follows:

Table 3: CDR/EDR export file name format

File name element	Length	Description
`<prefix>`	7	A fixed string. Always `sipwise`.
`<separator>`	1	A fixed character. Always `_`.
`<version>`	3	The format version, a three digit number. Currently `007`.
`<timestamp>`	14	The file creation timestamp in the format `YYYYMMDDhhmmss`.
`<sequence number>`	10	A unique 10-digit zero-padded sequence number for quick identification.
`<suffix>`	4	A fixed string. Always `.cdr` or `.edr`.

A valid example filename for a CDR billing file created at 2012-03-10 14:30:00 and being the 42nd file exported by the system, is:

`sipwise_007_20130310143000_0000000042.cdr`

9.2.2 File Format

Each billing file consists of three parts: one header line, zero to 5000 body lines and one trailer line.

File Header Format

The billing file header is one single line, which is constructed by the following fields:

`<version>,<number of records>`

The definition of the specific fields is as follows:

Table 4: CDR/EDR export file header line format

Body Element	Length	Type	Description
`<version>`	3	zero-padded uint	The format version. Currently `007`.
`<number of records>`	4	zero-padded uint	The number of body lines contained in the file.

A valid example for a Header is:

`007,0738`

File Body Format for Call Detail Records (CDR)

The body of a CDR consists of a minimum of zero and a maximum of 5000 lines. Each line holds one call detail record in CSV format and is constructed by the following fields, all of them enclosed in single quotes:

Table 5: CDR export file body line format

Body Element	Length	Type	Description
`<id>`	1-10	uint	Internal CDR id.

Table 5: (continued)

Body Element	Length	Type	Description
`<update_time>`	19	timestamp	Timestamp of last modification.
`<source_user_id>`	36	string	Internal UUID of calling party subscriber.
`<source_provider_id>`	1-255	string	Internal ID of calling party provider.
`<source_ext_subscriber_id>`	0-255	string	External ID of calling party subscriber.
`<source_subscriber_id>`	1-10	uint	Internal ID of calling party subscriber.
`<source_ext_account_id>`	0-255	string	External ID of calling party VoIP account.
`<source_account_id>`	1-10	uint	Internal ID of calling party VoIP account.
`<source_user>`	1-255	string	SIP username of calling party.
`<source_domain>`	1-255	string	SIP domain of calling party.
`<source_cli>`	1-64	string	CLI of calling party in E.164 format.
`<source_clir>`	1	uint	`1` for calls with CLIR, `0` otherwise.
`<source_ip>`	0-64	string	IP Address of the calling party.
`<destination_user_id>`	1 / 36	string	Internal UUID of called party subscriber or `0` if callee is not local.
`<destination_provider_id>`	1-255	string	Internal ID of called party provider.
`<dest_ext_subscriber_id>`	0-255	string	External ID of called party subscriber.
`<dest_subscriber_id>`	1-10	uint	Internal ID of called party subscriber.
`<dest_ext_account_id>`	0-255	string	External ID of called party VoIP account.
`<destination_account_id>`	1-10	uint	Internal ID of called party VoIP account.
`<destination_user>`	1-255	string	Final SIP username of called party.
`<destination_domain>`	1-255	string	Final SIP domain of called party.
`<destination_user_in>`	1-255	string	Incoming SIP username of called party.
`<destination_domain_in>`	1-255	string	Incoming SIP domain of called party.
`<dialed_digits>`	1-255	string	The user-part of the SIP Request URI as received by the soft-switch.
`<peer_auth_user>`	0-255	string	User to authenticate towards peer.
`<peer_auth_realm>`	0-255	string	Realm to authenticate towards peer.
`<call_type>`	3-4	string	The type of the call - one of: `call`: normal call `cfu`: call forward unconditional `cft`: call forward timeout `cfb`: call forward busy `cfna`: call forward no answer

Table 5: (continued)

Body Element	Length	Type	Description
`<call_status>`	2-7	string	The final call status - one of: `ok`: successful call `busy`: callee busy `noanswer`: no answer from callee `cancel`: cancel from caller `offline` callee offline `timeout`: no reply from callee `other`: unspecified, see `<call_code>` for details
`<call_code>`	3	uint	The final SIP status code.
`<init_time>`	23	timestamp	Timestamp of call initiation (invite received from caller). Seconds include fractional part (3 decimals).
`<start_time>`	23	timestamp	Timestamp of call establishment (final response received from callee). Seconds include fractional part (3 decimals).
`<duration>`	4-11	fixed precision	Length of call (beginning at `start_time`) in seconds with 3 decimals.
`<call_id>`	1-255	string	The SIP call-id.
`<rating_status>`	2-7	string	The internal rating status - one of: `unrated`: not rated `ok`: successfully rated `failed`: error while rating Currently always `ok` or `unrated`, depending on whether rating is enabled or not.
`<rated_at>`	0 / 19	timestamp	Timestamp of rating or empty if not rated.
`<source_carrier_cost>`	4-11	fixed precision	The originating carrier cost or empty if not rated. In cent with two decimals. `Only available in system exports, not for resellers.`
`<source_customer_cost>`	4-11	fixed precision	The originating customer cost or empty if not rated. In cent with two decimals.
`<source_carrier_zone>`	0-127	string	The originating carrier billing zone or empty if not rated. `Only available in system exports, not for resellers.`
`<source_customer_zone>`	0-127	string	The originating customer billing zone or empty if not rated.
`<source_carrier_destination>`	0-127	string	The originating carrier billing destination or empty if not rated. `Only available in system exports, not for resellers.`
`<source_customer_destination>`	0-127	string	The originating customer billing destination or empty if not rated.

Table 5: (continued)

Body Element	Length	Type	Description
`<source_carrier_free_time>`	1-10	uint	The number of originating free time seconds used on carrier side or empty if not rated. `Only available in system exports, not for resellers.`
`<source_customer_free_time>`	1-10	uint	The number of originating free time seconds used from the customer's account balance or empty if not rated.
`<destination_carrier_cost>`	4-11	fixed precision	The termination carrier cost or empty if not rated. In cent with two decimals. `Only available in system exports, not for resellers.`
`<destination_customer_cost>`	4-11	fixed precision	The termination customer cost or empty if not rated. In cent with two decimals.
`<destination_carrier_zone>`	0-127	string	The termination carrier billing zone or empty if not rated. `Only available in system exports, not for resellers.`
`<destination_customer_zone>`	0-127	string	The termination customer billing zone or empty if not rated.
`<destination_carrier_destination>`	0-127	string	The termination carrier billing destination or empty if not rated. `Only available in system exports, not for resellers.`
`<destination_customer_destination>`	0-127	string	The termination customer billing destination or empty if not rated.
`<destination_carrier_free_time>`	1-10	uint	The number of termination free time seconds used on carrier side or empty if not rated. `Only available in system exports, not for resellers.`
`<destination_customer_free_time>`	1-10	uint	The number of termination free time seconds used from the customer's account balance or empty if not rated.
`<source_reseller_cost>`	4-11	fixed precision	The originating reseller cost or empty if not rated. In cent with two decimals. `Only available in system exports, not for resellers.`
`<source_reseller_zone>`	0-127	string	The originating reseller billing zone or empty if not rated. `Only available in system exports, not for resellers.`
`<source_reseller_destination>`	0-127	string	The originating reseller billing destination or empty if not rated. `Only available in system exports, not for resellers.`
`<source_reseller_free_time>`	1-10	uint	The number of originating free time seconds used from the reseller's account balance or empty if not rated. `Only available in system exports, not for resellers.`
`<destination_reseller_cost>`	4-11	fixed precision	The termination reseller cost or empty if not rated. In cent with two decimals. `Only available in system exports, not for resellers.`

Table 5: (continued)

Body Element	Length	Type	Description
`<destination_reseller_zone>`	0-127	string	The termination reseller billing zone or empty if not rated. `Only available in system exports, not for resellers.`
`<destination_reseller_destination>`	0-127	string	The termination reseller billing destination or empty if not rated. `Only available in system exports, not for resellers.`
`<destination_reseller_free_time>`	1-10	uint	The number of termination free time seconds used from the reseller's account balance or empty if not rated. `Only available in system exports, not for resellers.`
`<line_terminator>`	1	string	A fixed character. Always `\n` (special char LF - ASCII `0x0A`).

A valid example of one body line of a rated CDR is (line breaks added for clarity):

```
'15','2013-03-26 22:09:11','a84508a8-d256-4c80-a84e-820099a827b0','1','','1','',
'2','testuser1','192.168.51.133','4311001','0','192.168.51.1',
'94d85b63-8f4b-43f0-b3b0-221c9e3373f2','1','','3','','4','testuser3',
'192.168.51.133','testuser3','192.168.51.133','testuser3','','','call','ok','200',
'2013-03-25 20:24:50.890','2013-03-25 20:24:51.460','10.880','44449842',
'ok','2013-03-25 20:25:27','0.00','24.00','onnet','testzone','platform internal',
'testzone','0','0','0.00','200.00','','foo','','foo','0','0',
'0.00','','','0','0.00','','','0'
```

The format of the CDR export files generated for resellers (as opposed to the complete system-wide export) is identical except for a few missing fields. Reseller CDR CSV files don't contain the fields for *carrier* or *reseller* ratings, neither in *source* nor *destination* direction. Thus, the reseller CSV files have 16 fewer fields.

File Body Format for Event Detail Records (EDR)

The body of a EDR consists of a minimum of zero and a maximum of 5000 lines. Each line holds one call detail record in CSV format and is constructed by the following fields, all of them enclosed in single quotes:

Table 6: EDR export file body line format

Body Element	Length	Type	Description
`<event_id>`	1-10	uint	Internal EDR id.

Table 6: (continued)

Body Element	Length	Type	Description
`<event_type>`	1-255	string	The type of the event - one of: `start_profile`: A subscriber profile has been newly assigned to a subscriber. `end_profile`: A subscriber profile has been removed from a subscriber. `update_profile`: A subscriber profile has been changed for a subscriber. `start_huntgroup`: A subscriber has been provisioned as group. `end_huntgroup`: A subscriber has been deprovisioned as group. `start_ivr`: A subscriber has a new call-forward to auto-attendant set. `end_ivr`: A subscriber has removed a call-forward to auto-attendant.
`<customer_external_id>`	0-255	string	The external customer ID as provisioned for the subscriber.
`<contact_company>`	0-255	string	The company name of the customer's contact.
`<subscriber_external_id>`	0-255	string	The external subscriber ID as provisioned for the subscriber.
`<subscriber_number>`	0-255	string	The voip number of the subscriber with the highest ID (DID or primary number).
`<old_status>`	0-255	string	The old status of the event. Depending on the event_type: `start_profile`: Empty. `end_profile`: The profile id of the profile which got removed from the subscriber. `update_profile`: The old profile id which got updated. `start_huntgroup`: Empty. `end_huntgroup`: The profile id of the group which got deprovisioned. `start_ivr`: Empty. `end_ivr`: Empty.

Table 6: (continued)

Body Element	Length	Type	Description
`<new_status>`	0-255	string	The new status of the event. Depending on the event_type: `start_profile`: The profile id which got assigned to the subscriber. `end_profile`: Empty. `update_profile`: The new profile id which got updated. `start_huntgroup`: The current profile id assigned to the group subscriber. `end_huntgroup`: The current profile id assigned to the group subscriber. `start_ivr`: Empty. `end_ivr`: Empty.
`<timestamp>`	0-255	string	The time when the event occured.
`<line_terminator>`	1	string	A fixed character. Always `\n` (special char LF - ASCII `0x0A`).

A valid example of one body line of an EDR is (line breaks added for clarity):

```
"1","start_profile","sipwise_ext_customer_id_4","Sipwise GmbH",
"sipwise_ext_subscriber_id_44","436667778","","1","2014-06-19 11:34:31"
```

File Trailer Format

The billing file trailer is one single line, which is constructed by the following fields:

```
<md5 sum>
```

The `<md5 sum>` is a 32 character hexadecimal MD5 hash of the *Header* and *Body*.

To validate the billing file, one must remove the Trailer before computing the MD5 sum of the file. An example bash script to validate the integrity of the file is given below:

```
#!/bin/sh

error() { echo $@; exit 1; }
test -n "$1" || error "Usage: $0 <cdr-file>"
test -f "$1" || error "File '$1' not found"

TMPFILE="/tmp/$(basename "$1").$$"
```

```
MD5="$(sed -rn '$ s/^([a-z0-9]{32}).*$/\1/i p' "$1")  $TMPFILE"
sed '$d' "$1" > "$TMPFILE"
echo "$MD5" | md5sum -c -
rm -f "$TMPFILE"
```

Given the script is located in `cdr-md5.sh` and the CDR-file is `sipwise_001_20071110123000_0000000004.cdr`,
the output of the integrity check for an intact CDR file would be:

```
$ ./cdr-md5.sh sipwise_001_20071110123000_0000000004.cdr
/tmp/sipwise_001_20071110123000_0000000004.cdr: OK
```

If the file has been altered during transmission, the output of the integrity check would be:

```
$ ./cdr-md5.sh sipwise_001_20071110123000_0000000004.cdr
/tmp/sipwise_001_20071110123000_0000000004.cdr: FAILED
md5sum: WARNING: 1 of 1 computed checksum did NOT match
```

9.2.3 File Transfer

Billing files are created twice per hour at minutes `25` and `55` and are stored in the home directory of the `cdrexport` user. If the
amount of records within the transmission interval exceeds the threshold of 5000 records per file, multiple billing files are created.
If no billing records are found for an interval, a billing file without body data is constructed for easy detection of lost billing files on
the 3rd party side.

CDR and EDR files are fetched by a 3rd party billing system using SFTP or SCP with either public key or password authentication
using the username cdrexport.
If public key authentication is chosen, the public key file has to be stored in the file `~/.ssh/authorized_keys2` below the
home directory of the `cdrexport` user. Otherwise, a password has to be set for the user.

The 3rd party billing system is responsible for deleting CDR files after fetching them.

Note

The `cdrexport` user is kept in a jailed environment on the system, so it has only access to a very limited set of commandline
utilities.

---

10   Provisioning interfaces

The sip:provider CE provides two kinds of provisioning interfaces for easy interconnection with 3rd party tools. The one recommended by Sipwise is the REST API, and the other (soon deprecated) one is SOAP and XMLRPC. Any new functionality is only added to the REST interface, so do not base any new development on SOAP or XMLRPC.

10.1   REST API

The sip:provider CE provides a REST API to provision various functionality of the platform. The entry point - and at the same time the official documentation - is at https://<your-ip>:1443/api. It allows both administrators and resellers (in a limited scope) to manage the system.

You can either authenticate via username and password of your administrative account you're using to access the admin panel, or via SSL client certificates. Find out more about client certificate authentication in the online api documentation.

10.2   SOAP and XMLRPC API

The sip:provider CE provides two (soon deprecated) XML based provisioning interfaces - SOAP and XMLRPC. The server provides online documentation about all the functions available. To access the online documentation for the first time, you need to follow the following instructions:

• Generate a password for http access to the provisioning interfaces:

```
htpasswd -nbs myuser mypassword
```

Note

Also see `man 1 htpasswd` on how to generate crypt or MD5 passwords if you like. Of course you may use any other process to generate crypt, MD5 or SHA hashed passwords. But using `htpasswd` ensures the hashes are also understood by Apache.

• Edit */etc/ngcp-config/config.yml*. Under section *ossbss→htpasswd*, replace *user* and *pass* with your new values and execute *ngcpcfg apply* as usual.

• Access https://<ip>:2443/SOAP/Provisioning.wsdl and login with your new credentials.

Note

The default port for provisioning interfaces is 2443. You can change it in */etc/ngcp-config/config.yml* by modifying *ossbss→apache→port* and execute *ngcpcfg apply*.

> **Important**
>
> The displayed online API documentation shows all the currently available functionalities. Enabling or disabling features in */etc/ngcp-config/config.yml* will directly reflect in the functions being available via the APIs.

> **Important**
>
> If your SOAP client throws errors because of the inline <documentation> tags (e.g. Visual Studio and the stock PHP SOAP client complain about this), try to use the WSDL URL https://<ip>:2443/SOAP/Provisioning.wsdl?plain instead, which supresses the output of these tags.

11   Configuration Framework

The sip:provider CE provides a configuration framework for consistent and easy to use low level settings management. A basic usage of the configuration framework only needs two actions already used in previous chapters:

- Edit */etc/ngcp-config/config.yml* file.

- Execute *ngcpcfg apply* command.

Low level management of the configuration framework might be required by advanced users though. This chapter explains the architecture and usage of the NGCP configuration framework. If the basic usage explained above fits your needs, feel free to skip this chapter and return to it when your requirements change.

A more detailed workflow of the configuration framework for creating a configuration file consists of 6 steps:

- Generation or editing of configuration templates and/or configuration values.

- Generation of the configuration files based on configuration templates and configuration values defined in config.yml, constants.yml and network.yml files.

- Execution of *prebuild* commands if defined for a particular configuration file or configuration directory.

- Placement of the generated configuration file in the target directory. This step is called *build* in the configuration framework.

- Execution of *postbuild* commands if defined for that configuration file or configuration directory.

- Execution of *services* commands if defined for that configuration file or configuration directory. This step is called *services* in the configuration framework.

- Saving of the generated changes. This step is called *commit* in the configuration framework.

11.1   Configuration templates

The sip:provider CE provides configuration file templates for most of the services it runs. These templates are stored in the directory */etc/ngcp-config/templates*.

Example: Template files for */etc/sems/sems.conf* are stored in */etc/ngcp-config/templates/etc/sems/*.

There are different types of files in this template framework, which are described below.

11.1.1   .tt2 and .customtt.tt2 files

These files are the main template files that will be used to generate the final configuration file for the running service. They contain all the configuration options needed for a running sip:provider CE system. The configuration framework will combine these files with the values provided by *config.yml*, *constants.yml* and *network.yml* to generate the appropriate configuration file.

Example: In the installation chapter we've changed the public interface from *lo* to *eth0*. This parameter will for example change kamailio's listen address, when the configuration file is generated. A quick look to the template file under */etc/ngcp-config/templates/etc/kama* will show a line like this:

```
listen=udp:[% ip %]:[% kamailio.lb.port %]
```

After applying the changes with the *ngcpcfg apply* command, a new configuration file will be created under */etc/kamailio/kamailio.cfg* with the proper values taken from the main configuration files (in this case *network.yml*):

```
listen=udp:1.2.3.4:5060
```

All the low-level configuration is provided by these .tt2 template files and the corresponding config.yml file. Anyways, advanced users might require a more particular configuration.

Instead of editing .tt2 files, the configuration framework recognises .customtt.tt2 files. These files are the same as .tt2, but they have higher priority when the configuration framework creates the final configuration files. An advanced user should create a .customtt.tt2 file from a copy of the corresponding .tt2 template and leave the .tt2 template untouched. This way, the user will have his personalized configuration and the system will continue providing a working, updated configuration template in .tt2 format.

Example: We'll create */etc/ngcp-config/templates/etc/kamailio.cfg.customtt.tt2* and use it for our personalized configuration. In this example, we'll just append a comment at the end of the template.

```
cd /etc/ngcp-config/templates/etc/kamailio/lb
cp kamailio.cfg.tt2 kamailio.cfg.customtt.tt2
echo '# This is my last line comment' >> kamailio.cfg.customtt.tt2
ngcpcfg apply
```

The ngcpcfg command will generate */etc/kamailio/kamailio.cfg* from our custom template instead of the general one.

```
tail -1 /etc/kamailio/kamailio.cfg
# This is my last line comment
```

Tip

The tt2 files use the Template Toolkit language. Therefore you can use all the feature this excellent toolkit provides within ngcpcfg's template files (all the ones with the .tt2 suffix).

11.1.2 .prebuild and .postbuild files

After creating the configuration files, the configuration framework can execute some commands before and after placing that file in its target directory. These commands usually are used for changing the file's owner, groups, or any other attributes. There are some rules these commands need to match:

- They have to be placed in a *.prebuild* or *.postbuild* file in the same path as the original *.tt2* file.

- The file name must be the same as the configuration file, but having the mentioned suffixes.

- The commands must be *bash* compatible.

- The commands must return 0 if successful.

- The target configuration file is matched by the environment variable *output_file*.

Example: We need *www-data* as owner of the configuration file */etc/ngcp-ossbss/provisioning.conf*. The configuration framework will by default create the configuration files with root:root as owner:group and with the same permissions (rwx) as the original template. For this particular example, we will change the owner of the generated file using the *.postbuild* mechanism.

```
echo 'chgrp www-data ${output_file}' \
  > /etc/ngcp-config/templates/etc/ngcp-ossbss/provisioning.conf.postbuild
```

11.1.3 .services files

.services files are pretty similar and might contain commands that will be executed after the *build* process. There are two types of *.services* files:

- The particular one, with the same name as the configuration file it is associated to.
  Example: */etc/ngcp-config/templates/etc/asterisk/sip.conf.services* is associated to */etc/asterisk/sip.conf*

- The general one, named *ngcpcfg.services* wich is associated to every file in its target directory.
  Example: */etc/ngcp-config/templates/etc/asterisk/ngcpcfg.services* is associated to every file under */etc/asterisk/*

When the *services* step is triggered all *.services* files associated to a changed configuration file will be executed. In case of the general file, any change to any of the configuration files in the directory will trigger the execution of the commands.

Tip

If the service script has the execute flags set (*chmod +x $file*) it will be invoked directly. If it doesn't have execute flags set it will be invoked under bash. Make sure the script is bash compatible if you do not set execute permissions on the service file.

These commands are usually service reload/restarts to ensure the new configuration has been loaded by running services.

Note

The configuration files mentioned in the following example usually already exist on the platform. Please make sure you don't overwrite any existing files if following this example.

Example:

```
echo '/etc/init.d/mysql restart' \
  > /etc/ngcpcfg-config/templates/etc/mysql/my.cnf.services
echo '/etc/init.d/asterisk restart' \
  > /etc/ngcpcfg-config/templates/etc/asterisk/ngcpcfg.services
```

In this example we created two *.services* files. Now, each time we trigger a change to */etc/mysql.my.cnf* or to */etc/asterisk/** we'll see that MySQL or Asterisk services will be restarted by the ngcpcfg system.

11.2   config.yml, constants.yml and network.yml files

The */etc/ngcp-config/config.yml* file contains all the user-configurable options, using the YAML (YAML Ain't Markup Language) syntax.

The */etc/ngcp-config/constants.yml* file provides configuration options for the platform that aren't supposed to be edited by the user. Do not manually edit this file unless you really know what you're doing.

The */etc/ngcp-config/network.yml* file provides configuration options for all interfaces and IP addresses on those interfaces. You can use the *ngcp-network* tool for conveniently change settings without having to manually edit this file.

The */etc/ngcp-config/ngcpcfg.cfg* file is the main configuration file for ngcpcfg itself. Do not manually edit this file unless you really know what you're doing.

11.3   ngcpcfg and its command line options

The ngcpcfg utility supports the following command line options:

11.3.1   apply

The *apply* option is a short-cut for the options "build && services && commit" and also executes *etckeeper* to record any modified files inside /etc. It is the recommended option to use the ngcpcfg framework unless you want to execute any specific commands as documented below.

11.3.2   build

The *build* option generates (and therefore also updates) configuration files based on their configuration (config.yml) and template files (.tt2). Before the configuration file is generated a present .prebuild will be executed, after generation of the configuration file the according .postbuild script (if present) will be executed. If a *file* or *directory* is specified as argument the build will generate only the specified configuration file/directory instead of running through all present templates.

Example: to generate only the file /etc/apache2/sites-available/ngcp-www-admin you can execute:

```
ngcpcfg build /etc/apache2/sites-available/ngcp-www-admin
```

Example: to generate all the files located inside the directory /etc/apache2/ you can execute:

```
ngcpcfg build /etc/apache2/
```

11.3.3   commit

The *commit* option records any changes done to the configuration tree inside */etc/ngcp-config*. The commit option should be executed when you've modified anything inside the configuration tree.

11.3.4 decrypt

Decrypt /etc/ngcp-config-crypted.tgz.gpg and restore configuration files, doing the reverse operation of the *encrypt* option. Note: This feature is only available if the ngcp-ngcpcfg-locker package is installed.

11.3.5 diff

Show uncommited changes between ngcpcfg's Git repository and the working tree inside */etc/ngcp-config*. Iff the tool doesn't report anything it means that there are no uncommited changes. If the *--addremove* option is specified then new and removed files (iff present) that are not yet (un)registered to the repository will be reported, no further diff actions will be executed then. Note: This option is available since ngcp-ngcpcfg version 0.11.0.

11.3.6 encrypt

Encrypt /etc/ngcp-config and all resulting configuration files with a user defined password and save the result as /etc/ngcp-config-crypted.tgz.gpg. Note: This feature is only available if the ngcp-ngcpcfg-locker package is installed.

11.3.7 help

The *help* options displays ngcpcfg's help screen and then exits without any further actions.

11.3.8 initialise

The *initialise* option sets up the ngcpcfg framework. This option is automatically executed by the installer for you, so you shouldn't have to use this option in normal operations mode.

11.3.9 pull

Retrieve modifications from shared storage. Note: This option is available in the High Availability setup only.

11.3.10 push

Push modifications to shared storage and remote systems. After changes have been pushed to the nodes the *build* option will be executed on each remote system to rebuild the configuration files (unless the --nobuild has been specified, then the build step will be skipped). If hostname(s) or IP address(es) is given as argument then the changes will be pushed to the shared storage and to the given hosts only. If no host has has been specified then the hosts specified in */etc/ngcp-config/systems.cfg* are used. Note: This option is available in the High Availability setup only.

11.3.11 services

The *services* option executes the service handlers for any modified configuration file(s)/directory.

11.3.12   status

The *status* option provides a human readable interface to check the state of the configuration tree. If you are unsure what should be done as next step or if want to check the current state of the configuration tree just invoke *ngcpcfg status*.

If everything is OK and nothing needs to be done the output should look like:

```
# ngcpcfg status
Checking state of ngcpcfg:
OK:   has been initialised already (without shared storage)
Checking state of configuration files:
OK:    nothing to commit.
Checking state of /etc files
OK:    nothing to commit.
```

If the output doesn't say "OK" just follow the instructions provided by the output of *ngcpcfg status*.

Further details regarding the ngcpcfg tool are available through *man ngcpcfg* on the Sipwise Next Generation Platform.

12   Network Configuration

Starting with version 2.7, the sip:provider CE uses a dedicated *network.yml* file to configure the IP addresses of the system. The reason for this is to be able to access all IPs of all nodes for all services from any particular node in case of a distributed system on one hand, and in order to be able the generate */etc/network/interfaces* automatically for all nodes based on this central configuration file.

12.1   General Structure

The basic structure of the file looks like this:

```
hosts:
  self:
    role:
      - proxy
      - lb
      - mgmt
    interfaces:
      - eth0
      - lo
    eth0:
      ip: 192.168.51.213
      netmask: 255.255.255.0
      type:
        - sip_ext
        - rtp_ext
        - web_ext
    lo:
      ip: 127.0.0.1
      netmask: 255.255.255.0
      type:
        - sip_int
        - web_int
        - ha_int
```

In CE systems, there is only one host entry in the file, and it's always named *self*.

12.2   Available Host Options

There are three different main sections for a host in the config file, which are *role*, *interfaces* and the actual interface definitions.

• *role*: The role setting is an array defining which logical roles a node will act as. Possible entries for this setting are:

  – *mgmt*: This entry means the host is acting as management node for the platform. In a sip:provider CE, this option must always been set. The management node exposes the admin and csc panels to the users and the APIs to external applications and

is used to export CDRs.

   – *lb*: This entry means the host is acting as SIP load-balancer for the platform. In a sip:provider CE, this option must always been set. The SIP load-balancer acts as an ingress and egress point for all SIP traffic to and from the platform.

   – *proxy*: This entry means the host is acting as SIP proxy for the platform. In a sip:provider CE, this option must always been set. The SIP proxy acts as registrar, proxy and application server and media relay, and is responsible for providing the features for all subscribers provisioned on it.

- *interfaces*: The interfaces setting is an array defining all interface names in the system. The actual interface details are set in the actual interface settings below.

- *<interface name>*: After the interfaces are defined in the *interfaces* setting, each of those interfaces needs to be specified as a separate setting with the following options:

   – *ip*

   – *netmask*

   – *advertised_ip*

   – *type*

There are different *interface types*, which define the services on a particular *interface*. For example the type *ssh_ext* set for a specific interface defines that the SSH daemon will listen on that interface for incoming connections. The list of possible types is as follows (note that you can assign a type only once per node):

- *mon_ext*: interface for monitoring purposes, e.g. for snmpd

- *rtp_ext*: interface for external RTP relay

- *sip_ext*: interface for external SIP communication between the sip:provider CE and the end points

- *sip_int*: interface for internal SIP communication, e.g. between load-balancer, proxy and application servers

- *ssh_ext*: interface for SSH remote login

- *web_ext*: interface for the administrative and customer web panels and the APIs

- *web_int*: interface for internal API communication

- *aux_ext*: interface for potentially insecure external components like rsyslogd service; e.g. the CloudPBX module can use those services to provide time services and remote logging facilities to end customer devices. The type *aux_ext* is assigned to *lo* interface by default. If it is needed to expose this type to the public, it is recommended to assign the type *aux_ext* to a separate VLAN interface to be able to limit or even block the incoming traffic easily via firewalling in case of emergency, like a (D)DOS attack on rsyslog services.

13 Security and Maintenance

Once the sip:provider CE is in production, security and maintenance becomes really important. In this chapter, we'll go through a set of best practices for any production system.

13.1 Firewalling

The sip:provider CE runs a wide range of services. Some of them need to interact with the user, while some others need to interact with the administrator or with nobody at all. Assuming that we trust the sip:provider CE server for outgoing connections, we'll focus only on incoming traffic to define the services that need to be open for interaction.

Table 7: Subscribers

Service	Default port	Config option
Customer self care interface	443 TCP	www_csc→apache→port
SIP	5060 UDP, TCP	kamailio→lb→port
SIP over TLS	5061 TCP	kamailio→lb→tls→port + kamailio→lb→tls→enable
RTP	30000-40000 UDP	rtpproxy→minport + rtpproxy→maxport
XCAP	1080 TCP	kamailio→proxy→presence→enable + nginx→xcap_port
XMPP	5222 and 5269 TCP	None, standard XMPP ports for clients (5222) and federation (5269)

Table 8: Administrators

Service	Default port	Config option
SSH/SFTP	22 TCP	NA
Administrator interface	1443 TCP	www_admin→apache→port
Provisioning interfaces	2443 TCP	ossbss→apache→port

> **Caution**
>
> To function correctly, the *mediaproxy* requires an additional *iptables* rule installed. This rule (with a target of `MEDIAPR`
> `OXY`) is automatically installed and removed when the mediaproxy starts and stops, so normally you don't need to worry
> about it. However, any 3rd party firewall solution can potentially flush out all existing iptables rules before installing
> its own, which would leave the system without the required `MEDIAPROXY` rule and this would lead to decreased
> performance. It is imperative that any 3rd party firewall solution either leaves this rule untouched, or installs it back into
> place after flushing all rules out. The complete parameters to install this rule (which needs to go into the `INPUT` chain
> of the `filter` table) are: `-p udp -j MEDIAPROXY --id 0`

13.2 Password management

The sip:provider CE comes with some default passwords the user should change during the deployment of the system. They have
been explained in the previous chapters of this document.

- The login for the system account *cdrexport* is disabled by default. Although this is a jailed account, it has access to sensitive
  information, namely the Call Detail Records of all calls. SSH keys should be used to login this user, or alternatively a really
  strong password should be used when setting the password via *passwd cdrexport*.

- The *root* user in MySQL has no default password. A password should be set using the *mysqladmin password* command.

- The administrative web interface has a default user *administrator* with password *administrator*. It should be changed within this
  interface.

- Generate new password for user *ngcpsoap* to access the provisioning interfaces, see the details in Section 10.

The Vagrant/VirtualBox/VMWare sip:provider CE images come with more default credentials which should be changed immedi-
ately:

- The default password of the system account *root* is *sipwise*. A password must be changed immediately using command *passwd
  root*.

- SSH authorized_keys for users *root* and *sipwise* should be wiped out using command *rm ~root/.ssh/authorized_keys ~sip-
  wise/.ssh/authorized_keys* for VirtualBox/VMWare images (skip the step if you use Vagrant).

> **Important**
>
> Many NGCP services use MySQL backend. Users and passwords for these services are created during the installation.
> These passwords are unique for each installation, and the connections are restricted to localhost. You should not
> change these users and passwords.

13.3 SSL certificates.

The sip:provider CE provides default, self-signed SSL certificates for SSL connections. These certificates are common for every
installation. Before going to production state, the system administrator should provide SSL certificates for the web services. These

certificates can either be shared by all web interfaces (*provisioning*, *administrator interface* and *customer self care interface*), or separate ones for each them can be used.

• Generate the certificates. The *customer self care interface* certificate should be signed by a certification authority to avoid browser warnings.

• Upload the certificates to the system

• Set the path to the new certificates in */etc/ngcp-config/config.yml*:

  – *ossbss→apache→sslcertfile* and *ossbss→apache→sslcertkeyfile* for the *provisioning interface*.

  – *www_admin→apache→sslcertfile* and *www_admin→apache→sslcertkeyfile* for the *admin interface*.

  – *www_csc→apache→sslcertfile* and *www_csc→apache→sslcertkeyfile* for the *customer self care interface*.

• Apply the configuration changes with *ngcpcfg apply*.

The sip:provider CE also provides the self-signed SSL certificates for SIP over TLS services. The system administrator should replace them with certificates signed by a trusted certificate authority if he is going to enable it for the production usage (*kamailio→lb→tls→enable* (disabled by default)).

• Generate the certificates.

• Upload the certificates to the system

• Set the path to the new certificates in */etc/ngcp-config/config.yml*:

  – *kamailio→lb→tls→sslcertfile* and *kamailio→lb→tls→sslcertkeyfile* .

• Apply the configuration changes with *ngcpcfg apply*.

13.4   Backup and recovery

13.4.1   Backup

The sip:provider CE can be integrated with most of the existing backup solutions. While it does not provide any backup system by default, any Debian compatible system can be installed. It's not the scope of this chapter to go through backup system configuration. We'll focus on which information needs to be saved.

The minimum set of information to be backed up is:

• The database information.

This is the most important data in the system. All subscriber information, billing, CDRs, user preferences etc. are stored in the MySQL server. A periodical dump of all the databases should be performed.

• System configuration options

/etc/ngcp-config/config.yml, */etc/ngcp-config/constants.yml*, */etc/ngcp-config/network.yml,* /etc/mysql/debian.cnf *and* /etc/mysql/sip-wise.cnf *files, where your specific system configurations are stored, should be included in the backup as well. Saving the entire* /etc/ngcp-config__ folder is a good idea in general.

- Optional: Exported CDRs

The directory */home/jail/home/cdrexport* contains the exported CDRs the system has generated so far. It depends on your local call data retention policy whether or not to remove these files after exporting them to an external system.

- Optional: Custom files

Any custom configurations, like modified templates or additionally implemented services which are not provided by the sip:provider CE

13.4.2 Recovery

In the worst case scenario, when the system needs to be recovered from a total loss, you only need 4 steps to get back online:

- Install the sip:provider CE as explained in chapter 2.

- Restore *config.yml*, *constants.yml*, *debian.cnf* and *sipwise.cnf* from the backup, overwriting your local files.

- Restore the database dump.

- Execute *ngcpcfg apply*.

13.5 Reset database

To reset database to its original state you can use the script provided by CE: * Execute *ngcp-reset-db*. It will assign new unique password for the NGCP services and restart all services. IMPORTANT: All existing data will be wiped out without possibility of restoring.

13.6 System requirements and performance

The sip:provider CE is a very flexible system, capable of serving from hundreds to several tens of thousands of subscribers in a single node. The system comes with a default configuration, capable of serving up to 50.000 subscribers in a *normal* environment. But there is no such thing as a *normal* environment. And the sip:provider CE has sometimes to be tunned for special environments, special hardware requirements or just growing traffic.

Note

If you have performance issues with regards to disk I/O please consider enabling the *noatime* mount option for the root filesystem. Sipwise recommends the usage of *noatime*, though remove it if you use software which conflicts with its presence.

In this section some parameters will be explained to allow the sip:provider CE administrator tune the system requirements for optimum performance.

Table 9: Requirement_options

Option	Default value	Requirement impact
cleanuptools→binlog_days	15	Heavy impact on the harddisk storage needed for mysql logs. It can help to restore the database from backups or restore broken replication.
database→bufferpoolsize	64MB	For test systems or low RAM systems, lowering this setting is one of the most effective ways of releasing RAM. The administrator can check the innodb buffer hit rate on production systems; a hit rate over 99% is desired to avoid bottlenecks.
kamailio→lb→pkg_mem	16	This setting affects the amount of RAM the system will use. Each kamailio-lb worker will have this amount of RAM reserved. Lowering this setting up to 8 will help to release some memory depending on the number of kamailio-lb workers running. This can be a dangerous setting as the lb process could run out of memory. Use with caution.
kamailio→lb→shm_mem	1/16 * Total System RAM	The installer will set this value to 1/16 of the total system RAM. This setting does not change even if the system RAM does so it's up to the administrator to tune it. It has been calculated that 1024 (1GB) is a good value for 50K subscriber environment. For a test environment, setting the value to 64 should be enough. "Out of memory" messages in the kamailio log can indicate that this value needs to be raised.
kamailio→lb→tcp_children	8	Number of TCP workers kamailio-lb will spawn per listening socket. The value should be fine for a mixed UDP-TCP 50K subscriber system. Lowering this setting can free some RAM as the number of kamailio processess would decrease. For a test system or a pure UDP subscriber system 2 is a good value. 1 or 2 TCP workers are always needed.
kamailio→lb→tls→enable	yes	Enable or not TLS signaling on the system. Setting this value to "no" will prevent kamailio to spawn TLS listening workers and free some RAM.
kamailio→lb→udp_children	8	See *kamailio→lb→tcp_children* explanation
kamailio→proxy→children	8	See *kamailio→lb→tcp_children* explanation. In this case the proxy only listens udp so these children should be enough to handle all the traffic. It could be set to 2 for test systems to lower the requirements.
kamailio→proxy→*_expires		Set the default and the max and min registration interval. The lower it is more REGISTER requests will be handled by the lb and the proxy. It can impact in the network traffic, RAM and CPU usage.
kamailio→proxy→natping_interval	30	Interval for the proxy to send a NAT keepalive OPTIONS message to the nated subscriber. If decreased, this setting will increase the number of OPTIONS requests the proxy needs to send and can impact in the network traffic and the number of natping processes the system needs to run. See *kamailio→proxy→natping_processes* explanation.

Table 9: (continued)

Option	Default value	Requirement impact
kamailio→proxy→natping_processes	7	Kamailio-proxy will spawn this number of processes to send keepalive OPTIONS to the nated subscribers. Each worker can handle about 250 messages/second (depends on the hardware). Depending the number of nated subscribers and the *kamailio→proxy→natping_interval* parameter the number of workers may need to be adjusted. The number can be calculated like nated_subscribers/natping_interval/pings_per_second_per_process. For the default options, asuming 50K nated subscribers in the system the parameter value would be 50.000/30/250 = (6,66) 7 workers. 7 is the maximum number of processes kamailio will accept. Raising this value will cause kamailio not to start.
kamailio→proxy→shm_mem	1/16 * Total System RAM	See *kamailio→lb→shm_mem* explanation.
rateomat→enable	yes	Set this to no if the system shouldn't perform rating on the CDRs. This will save CPU usage.
rsyslog→external_log	0	If enabled, the system will send the log messages to an external server. Depending on the *rsyslog→external_loglevel* parameter this can increase dramatically the network traffic.
rsyslog→ngcp_logs_preserve_days	93	This setting will set the number of days ngcp logs under */var/log/ngcp* will be kept in disk. Lowering this setting will free a high amount of disk space.